1/35

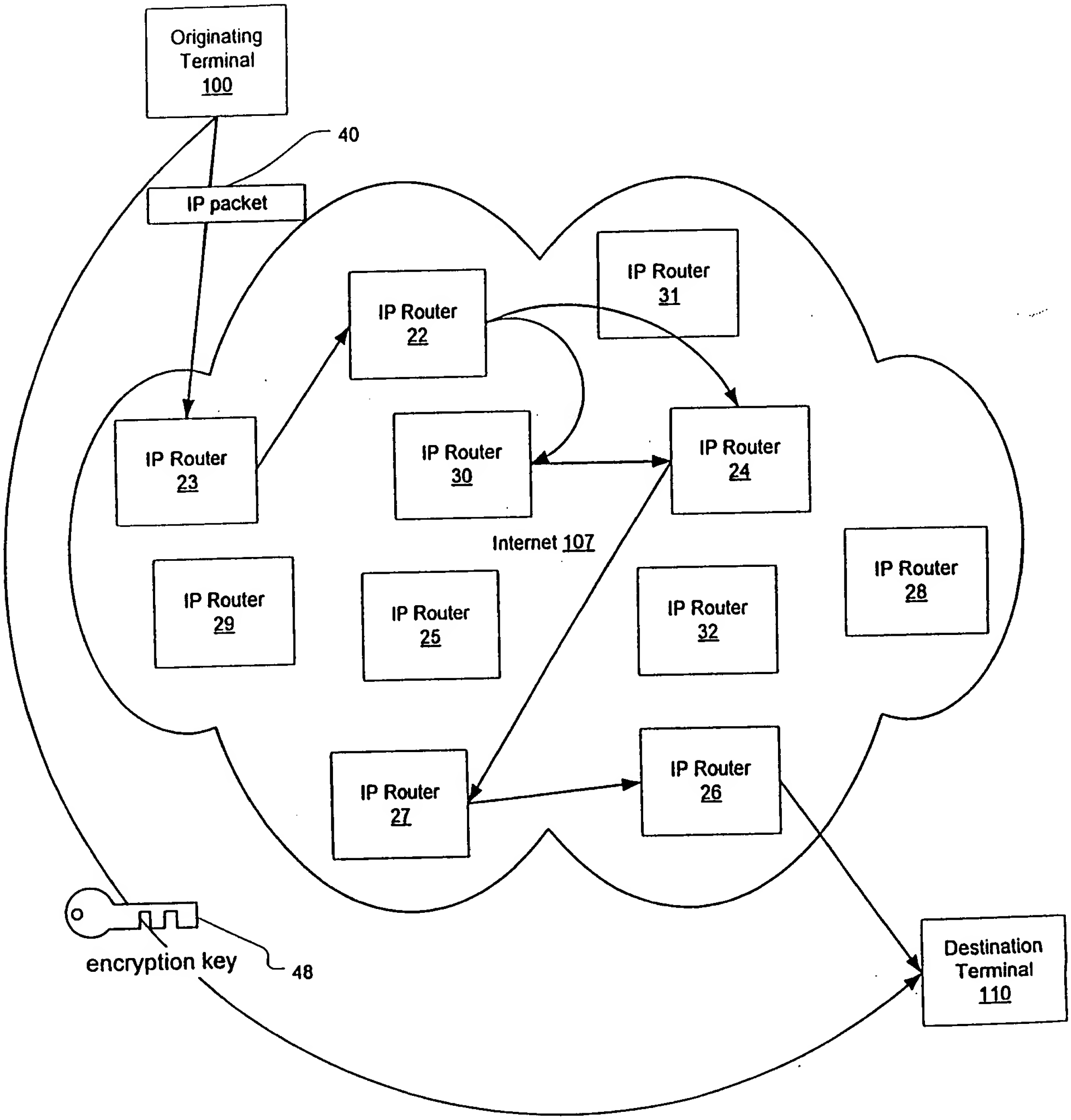

Originating Terminal 100
40
IP packet
IP Router 31
IP Router 22
IP Router 23
IP Router 30
IP Router 24
Internet 107
IP Router 28
IP Router 29
IP Router 25
IP Router 32
IP Router 27
IP Router 26
encryption key
48
Destination Terminal 110

Fig. 1

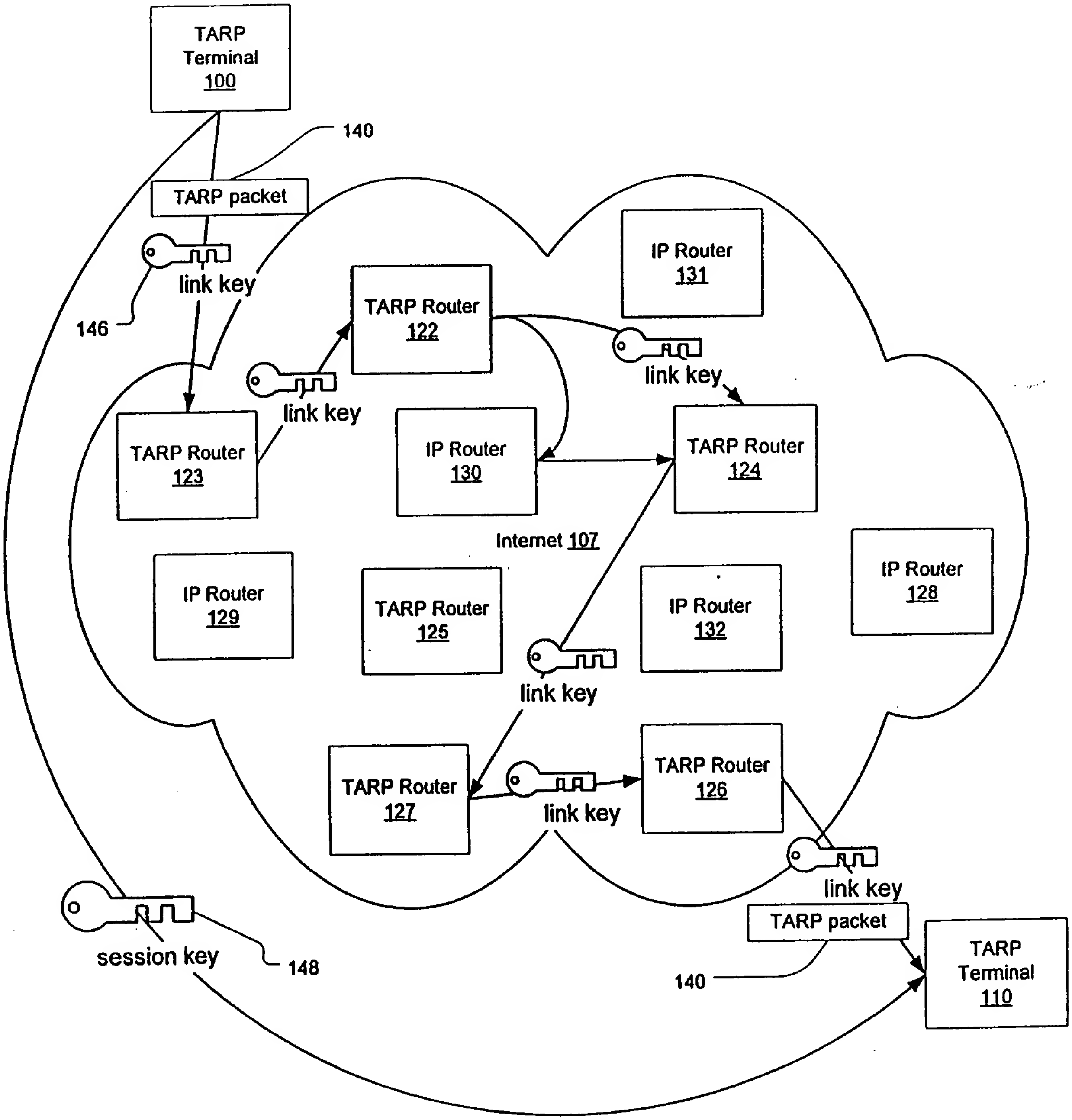

TARP Terminal 100
140
TARP packet
link key
146
TARP Router 122
link key
IP Router 131
link key
TARP Router 123
IP Router 130
TARP Router 124
Internet 107
IP Router 129
TARP Router 125
IP Router 132
IP Router 128
link key
TARP Router 127
link key
TARP Router 126
link key
TARP packet
session key
148
140
TARP Terminal 110

Fig. 2

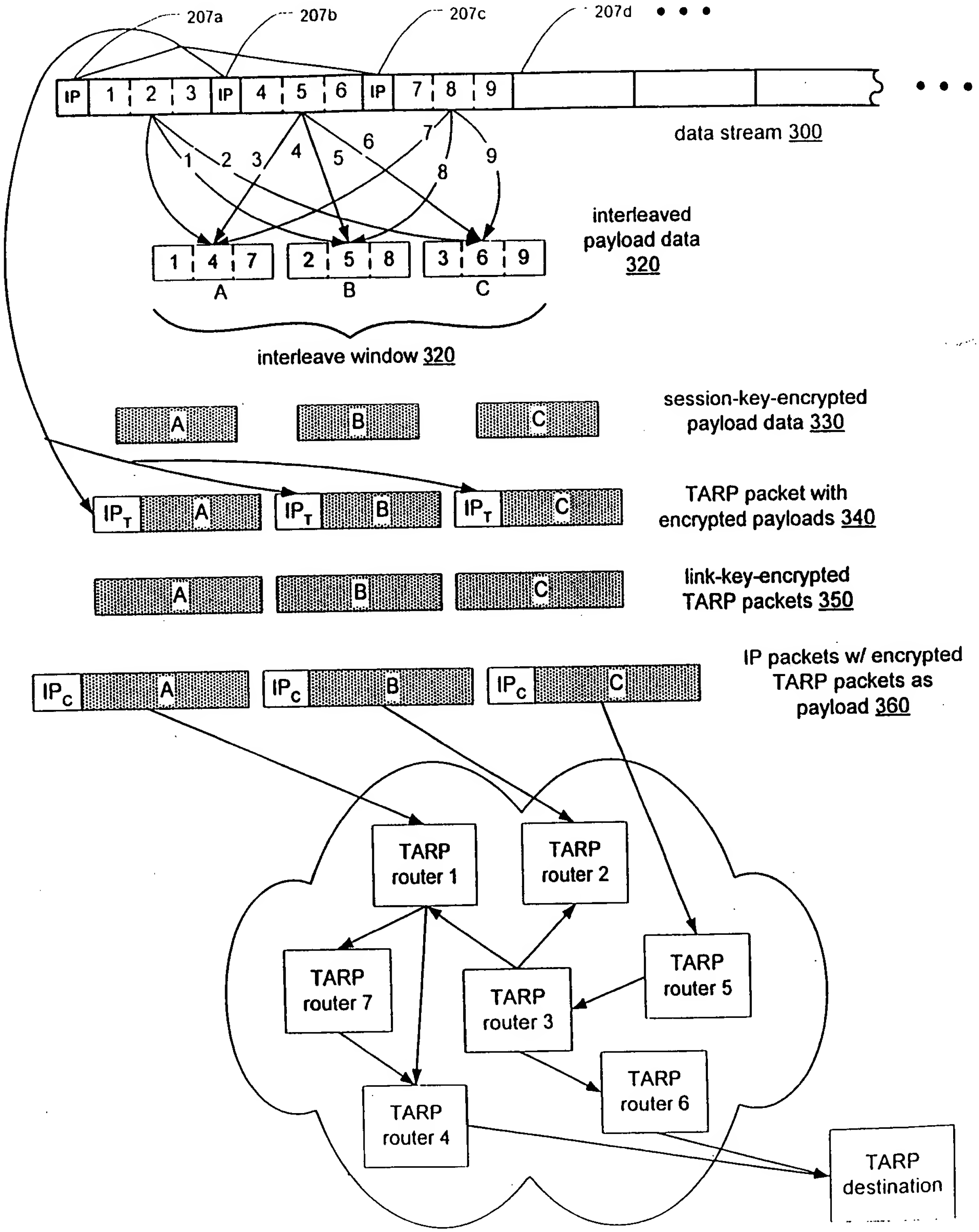

207a
207b
207c
207d
IP
1
2
3
IP
4
5
6
IP
7
8
9
data stream 300
1
2
3
4
5
6
7
8
9
1
4
7
2
5
8
3
6
9
A
B
C
interleaved payload data 320
interleave window 320
A
B
C
session-key-encrypted payload data 330
IP_T
A
IP_T
B
IP_T
C
TARP packet with encrypted payloads 340
A
B
C
link-key-encrypted TARP packets 350
IP_C
A
IP_C
B
IP_C
C
IP packets w/ encrypted TARP packets as payload 360
TARP router 1
TARP router 2
TARP router 7
TARP router 3
TARP router 5
TARP router 6
TARP router 4
TARP destination

Fig. 3a

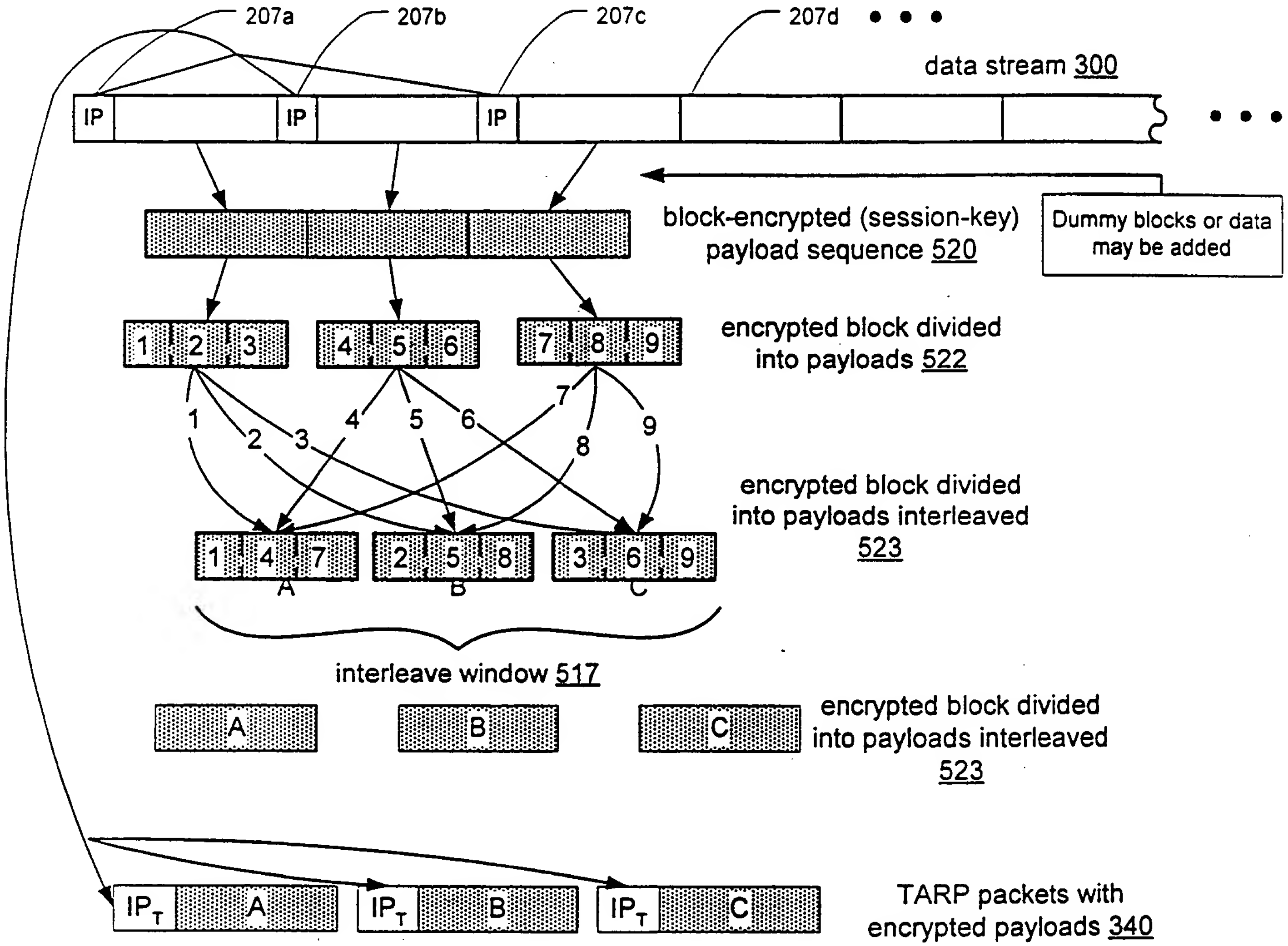

207a
207b
207c
207d
data stream 300
IP
IP
IP
block-encrypted (session-key) payload sequence 520
Dummy blocks or data may be added
encrypted block divided into payloads 522
1
2
3
4
5
6
7
8
9
encrypted block divided into payloads interleaved 523
A
B
C
interleave window 517
encrypted block divided into payloads interleaved 523
TARP packets with encrypted payloads 340
IP_T

Fig. 3b

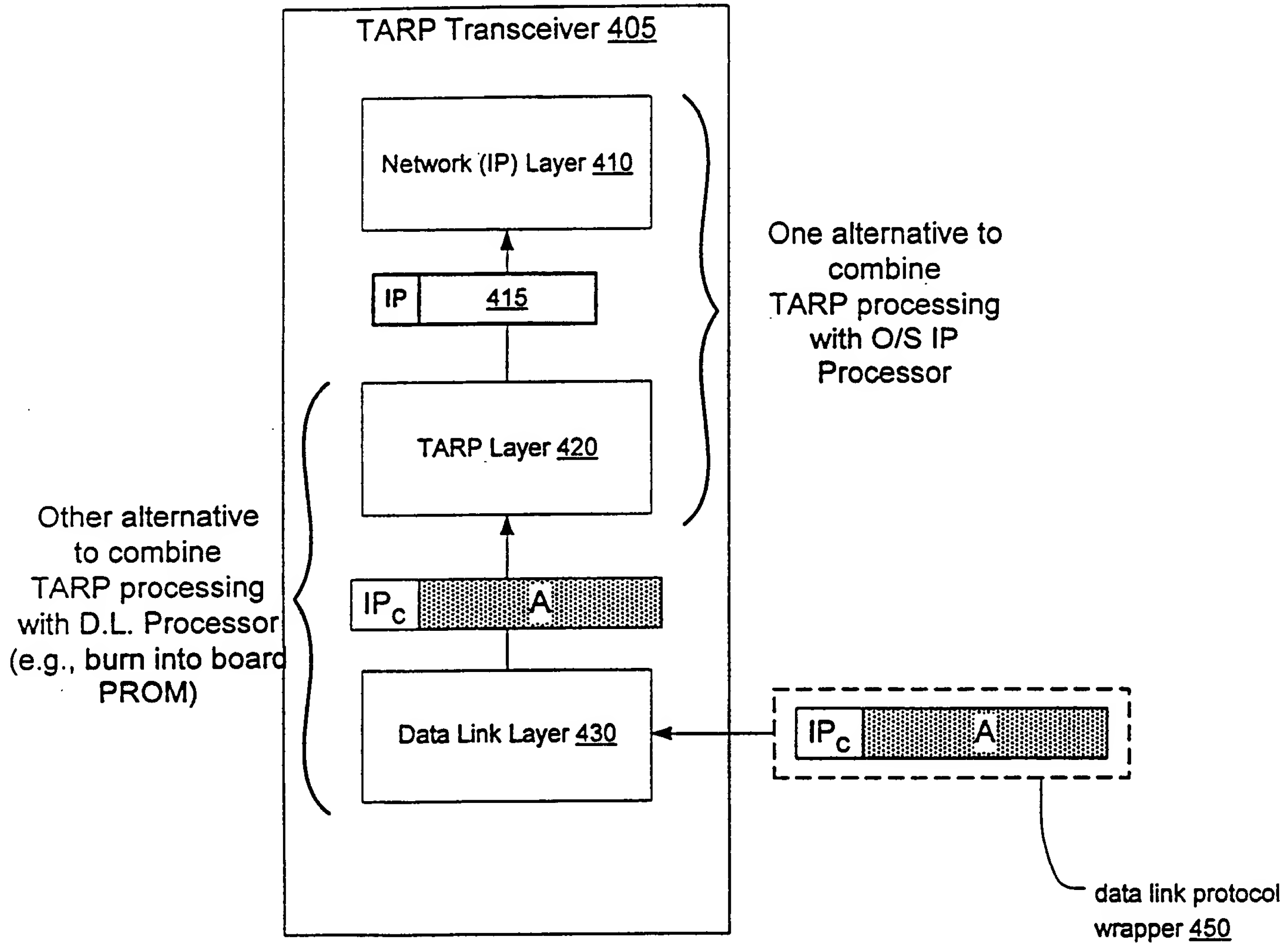
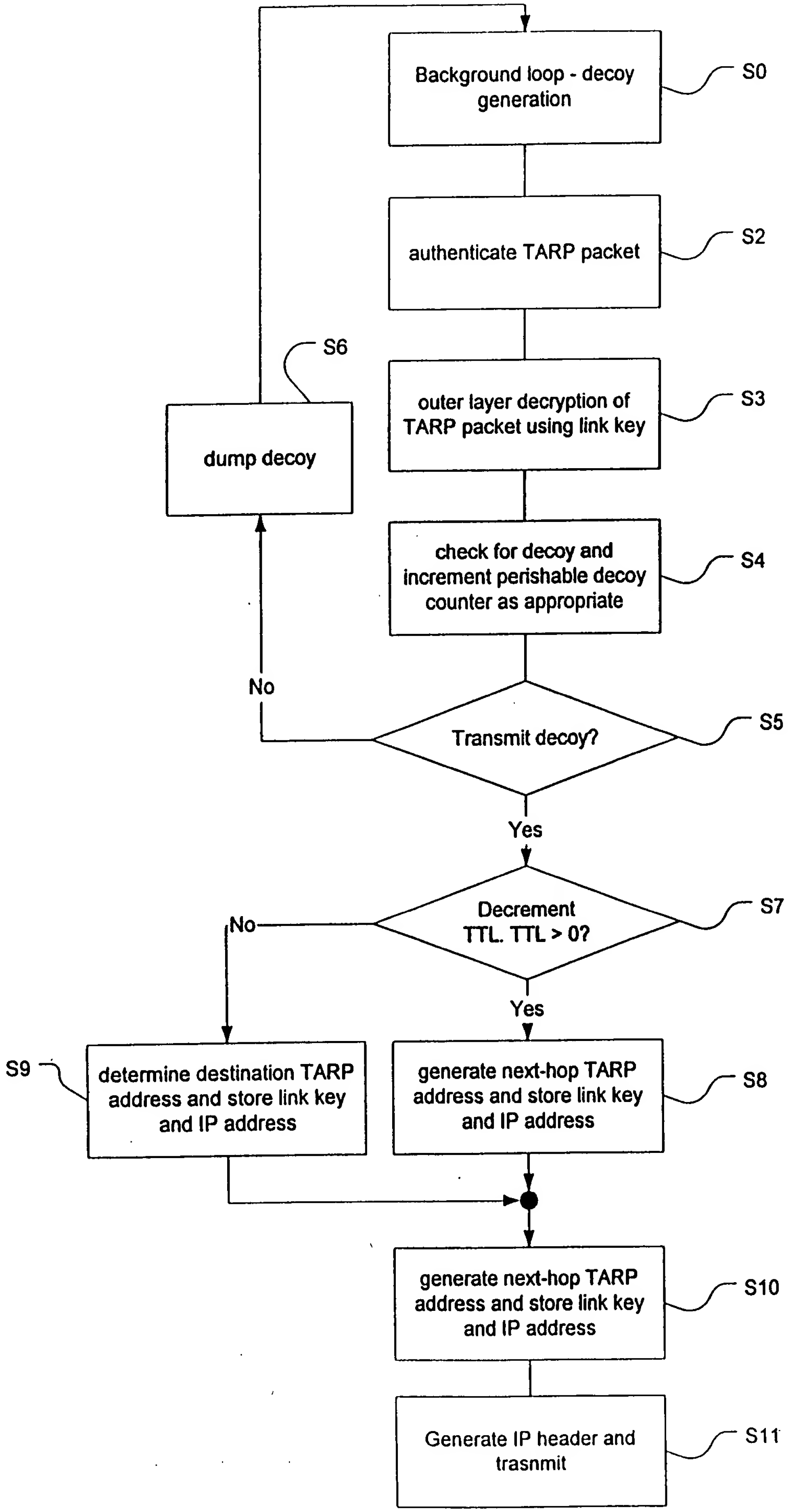

TARP Transceiver 405
Network (IP) Layer 410
IP
415
TARP Layer 420
IPc
A
Data Link Layer 430
One alternative to combine TARP processing with O/S IP Processor
Other alternative to combine TARP processing with D.L. Processor (e.g., burn into board PROM)
IPc
A
data link protocol wrapper 450

Fig. 4



Background loop - decoy generation
S0
authenticate TARP packet
S2
S6
outer layer decryption of TARP packet using link key
S3
dump decoy
check for decoy and increment perishable decoy counter as appropriate
S4
No
Transmit decoy?
S5
Yes
Decrement TTL. TTL > 0?
S7
No
Yes
S9
determine destination TARP address and store link key and IP address
generate next-hop TARP address and store link key and IP address
S8
generate next-hop TARP address and store link key and IP address
S10
Generate IP header and trasnmit
S11

Fig. 5

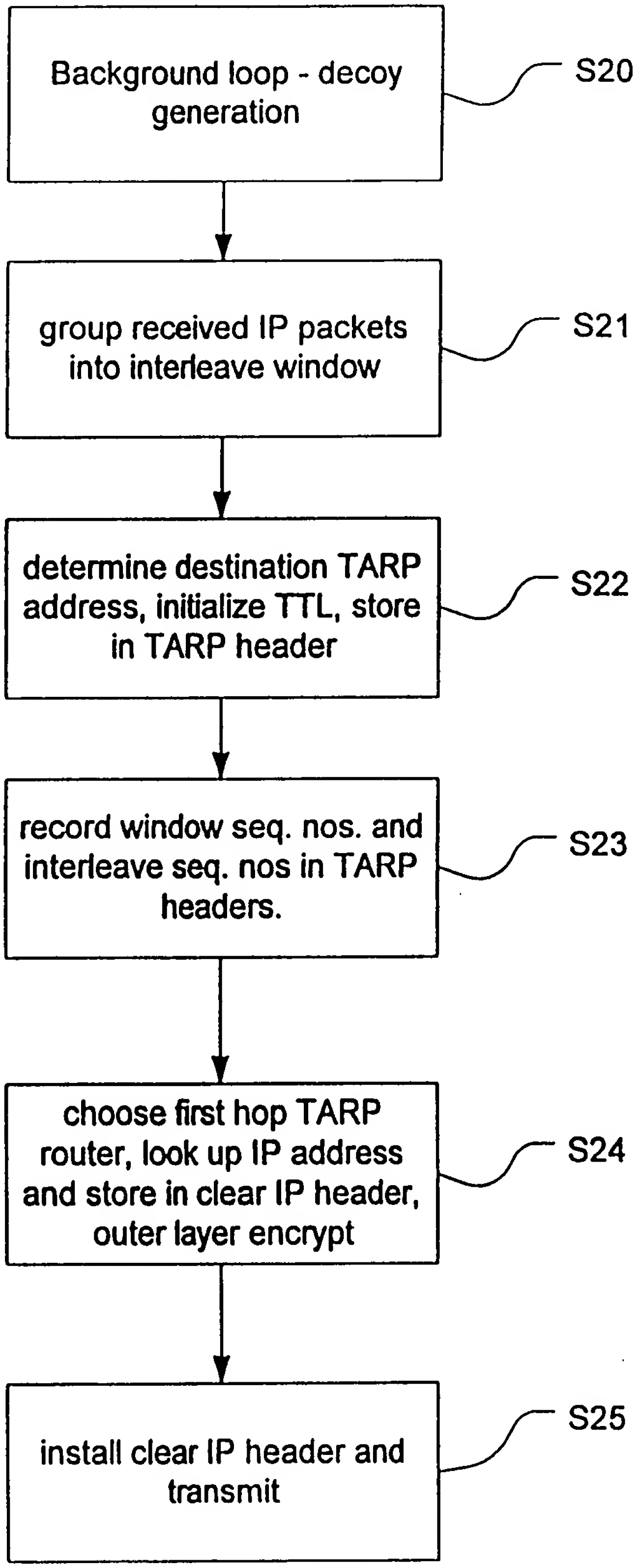

Background loop - decoy generation
S20
group received IP packets into interleave window
S21
determine destination TARP address, initialize TTL, store in TARP header
S22
record window seq. nos. and interleave seq. nos in TARP headers.
S23
choose first hop TARP router, look up IP address and store in clear IP header, outer layer encrypt
S24
install clear IP header and transmit
S25

Fig. 6

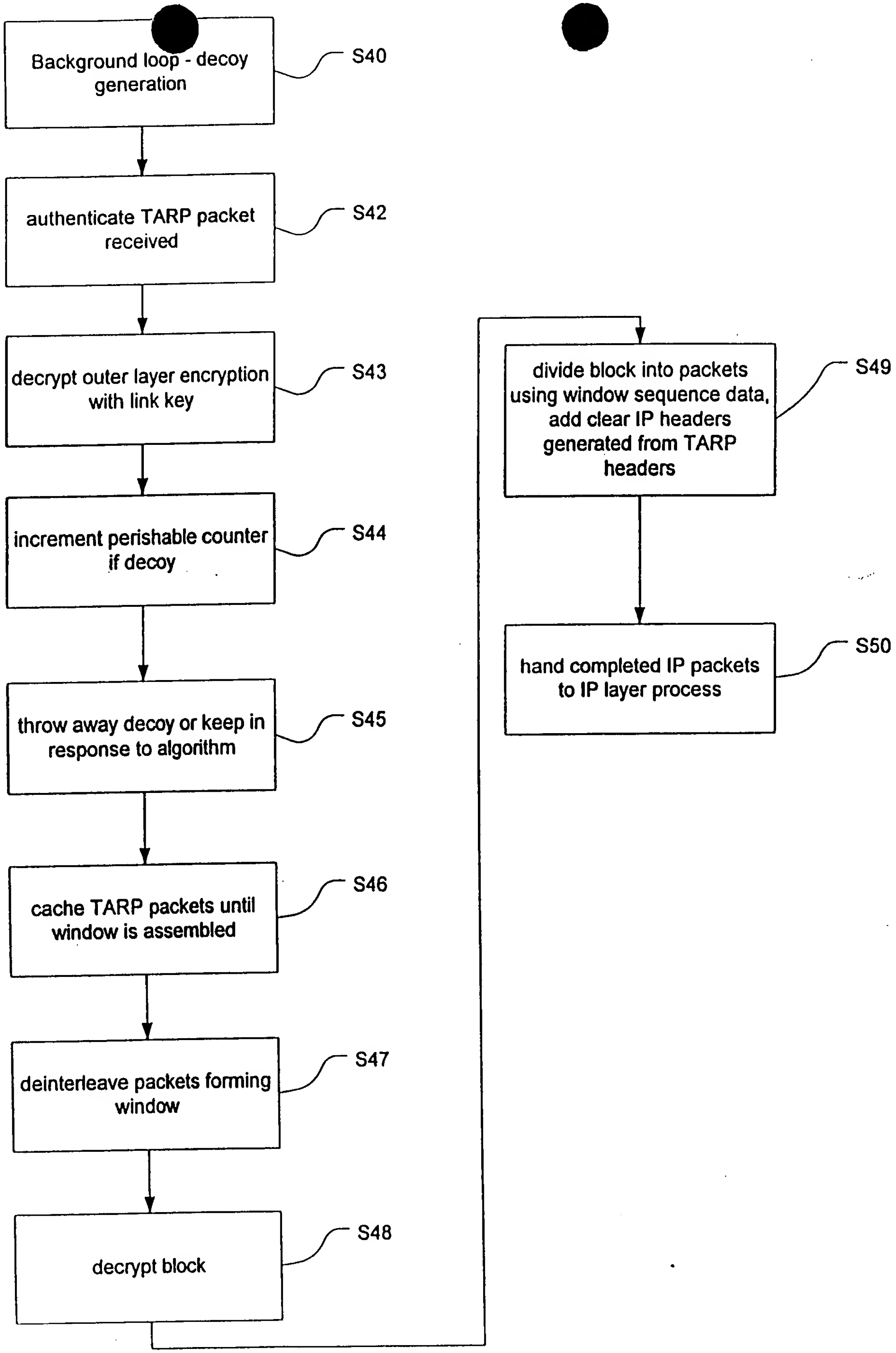

Background loop - decoy generation
S40
authenticate TARP packet received
S42
decrypt outer layer encryption with link key
S43
increment perishable counter if decoy
S44
throw away decoy or keep in response to algorithm
S45
cache TARP packets until window is assembled
S46
deinterleave packets forming window
S47
decrypt block
S48
divide block into packets using window sequence data, add clear IP headers generated from TARP headers
S49
hand completed IP packets to IP layer process
S50

Fig. 7

FIG. 8

SECURE SESSION ESTABLISHMENT
AND SYNCHRONIZATION

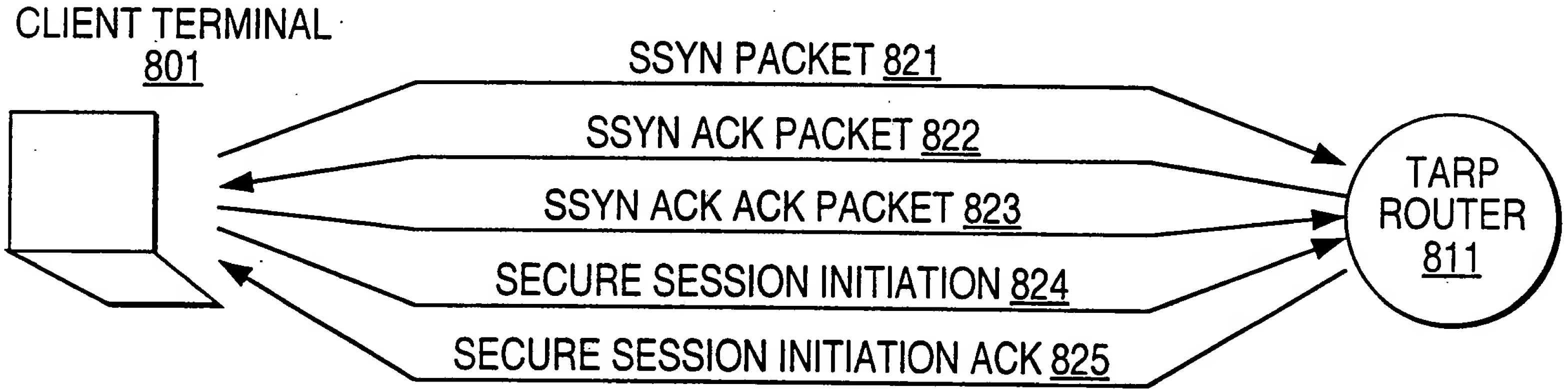
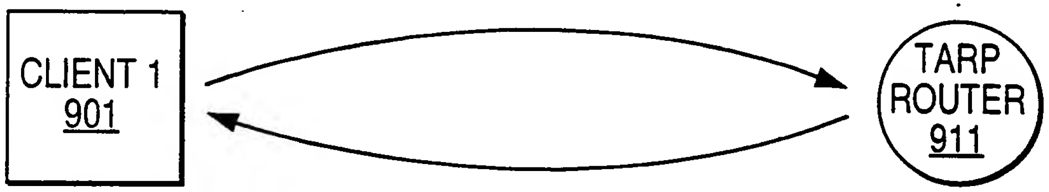


FIG. 9

IHOP TRANSMIT AND RECEIVE TABLES



TRANSMIT TABLE 921

131.218.204.98	,	131.218.204.65
131.218.204.221	,	131.218.204.97
131.218.204.139	,	131.218.204.186
131.218.204.12	,	131.218.204.55
⋮		⋮

RECEIVE TABLE 924

131.218.204.98	,	131.218.204.65
131.218.204.221	,	131.218.204.97
131.218.204.139	,	131.218.204.186
131.218.204.12	,	131.218.204.55
⋮		⋮

RECEIVE TABLE 922

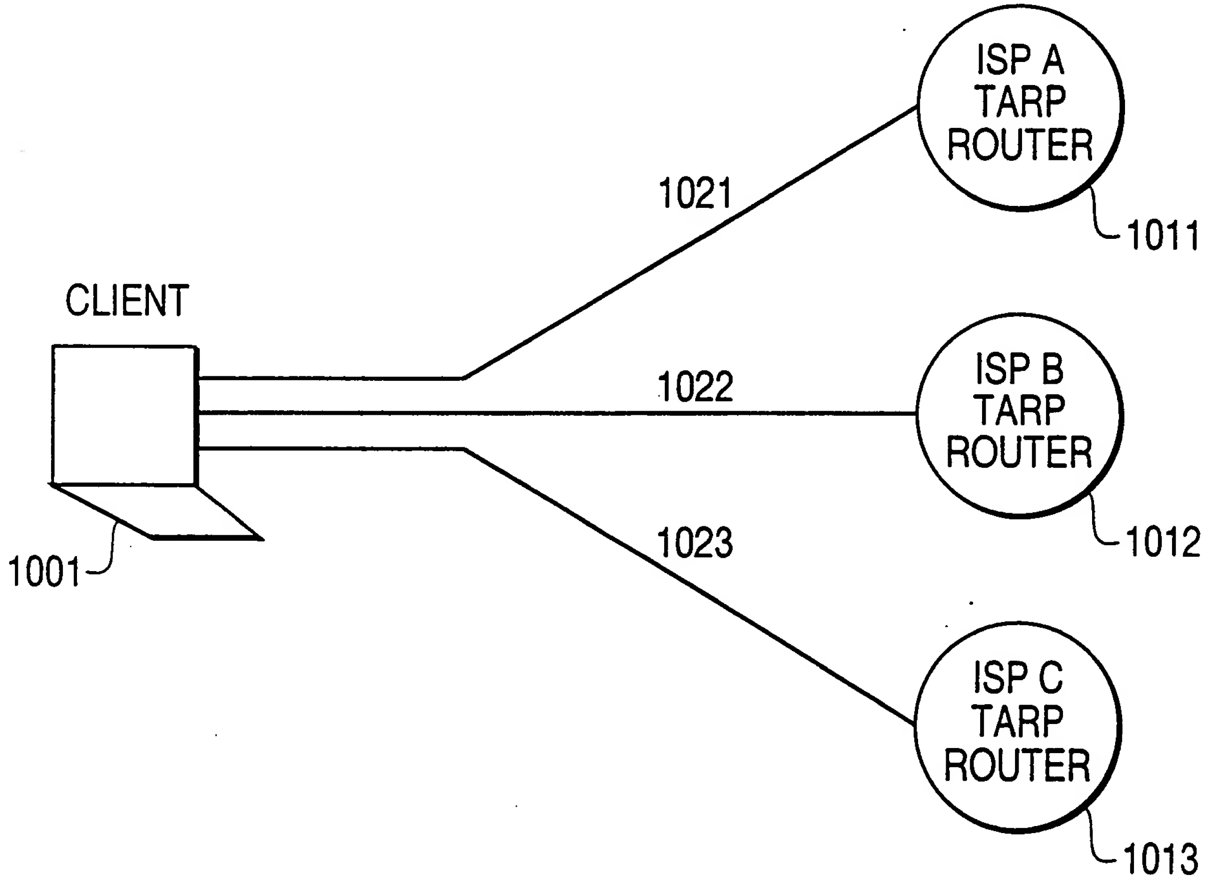
131.218.204.161	,	131.218.204.89
131.218.204.66	,	131.218.204.212
131.218.204.201	,	131.218.204.127
131.218.204.119	,	131.218.204.49
⋮		⋮

TRANSMIT TABLE 923

131.218.204.161	,	131.218.204.89
131.218.204.66	,	131.218.204.212
131.218.204.201	,	131.218.204.127
131.218.204.119	,	131.218.204.49
⋮		⋮

FIG. 10

PHYSICAL LINK REDUNDANCY



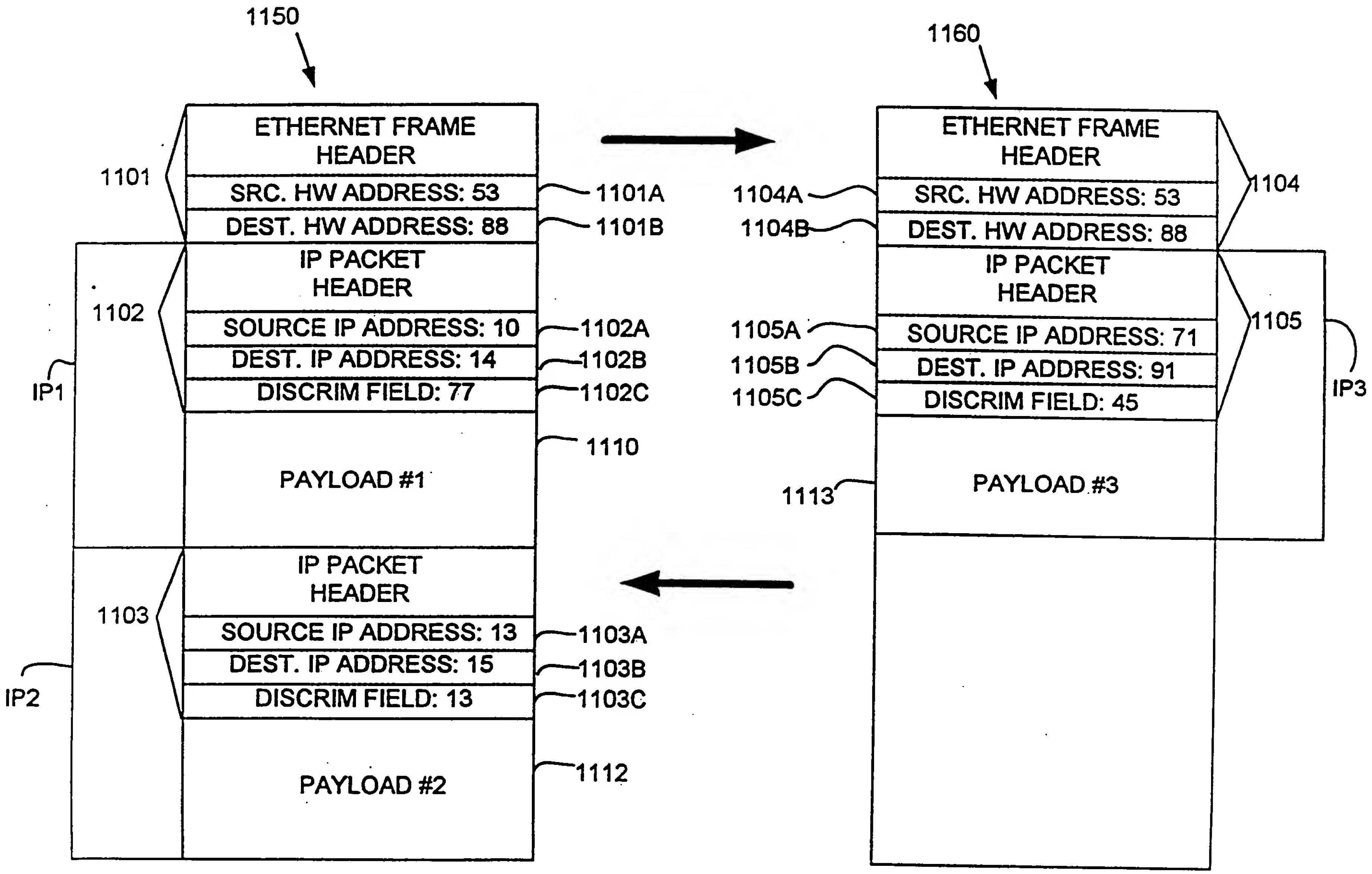
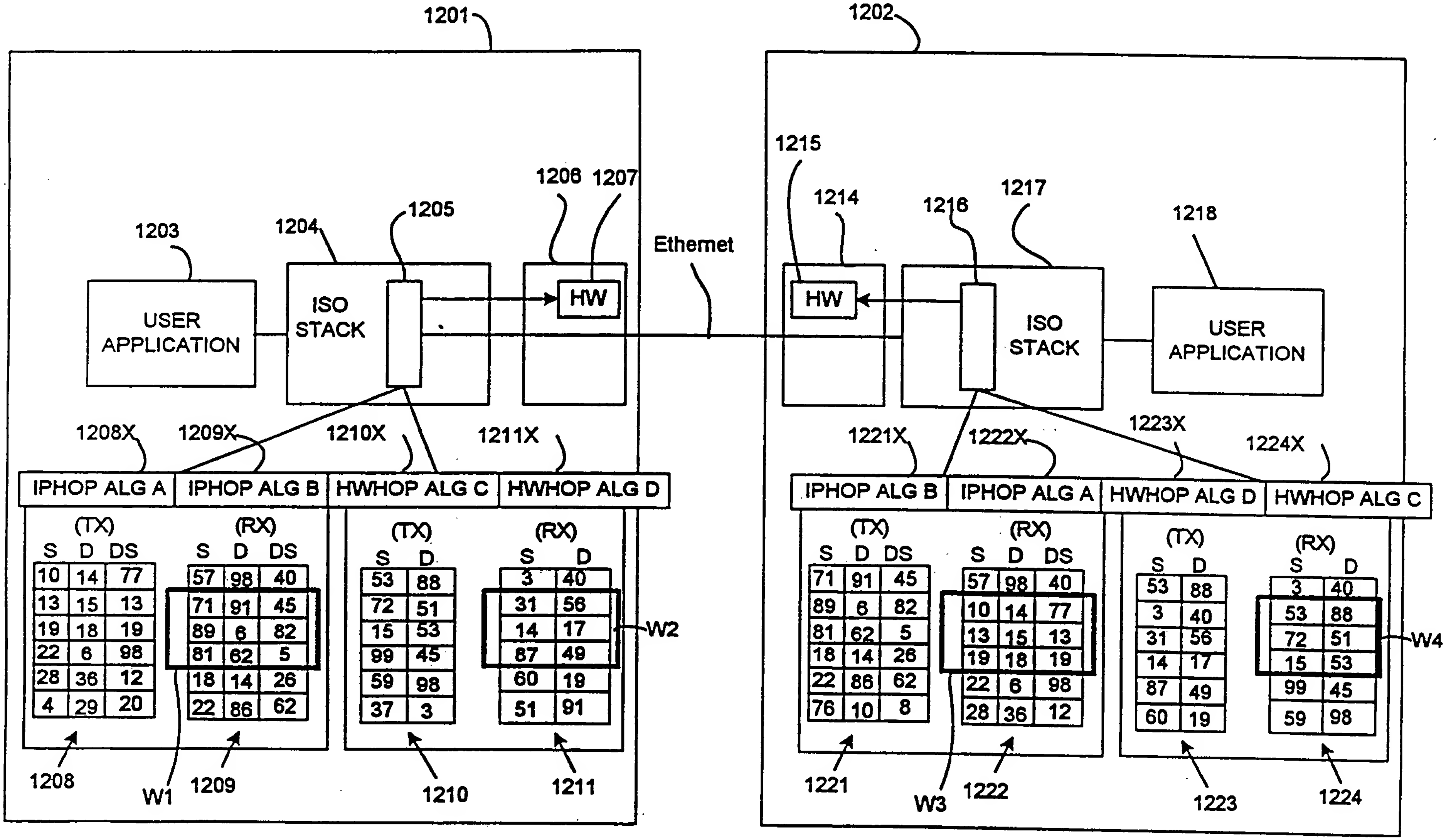

1150
1160
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
PAYLOAD #1
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
PAYLOAD #2
1101
1102
1103
IP1
IP2
1101A
1101B
1102A
1102B
1102C
1110
1103A
1103B
1103C
1112
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
PAYLOAD #3
1104A
1104B
1105A
1105B
1105C
1113
1104
1105
IP3

FIG. 11



(TX)

S	D	DS
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12
4	29	20

(RX)

S	D	DS
57	98	40
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62

(TX)

S	D
53	88
72	51
15	53
99	45
59	98
37	3

(RX)

S	D
3	40
31	56
14	17
87	49
60	19
51	91

(TX)

S	D	DS
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62
76	10	8

(RX)

S	D	DS
57	98	40
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12

(TX)

S	D
53	88
3	40
31	56
14	17
87	49
60	19

(RX)

S	D
3	40
53	88
72	51
15	53
99	45
59	98

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

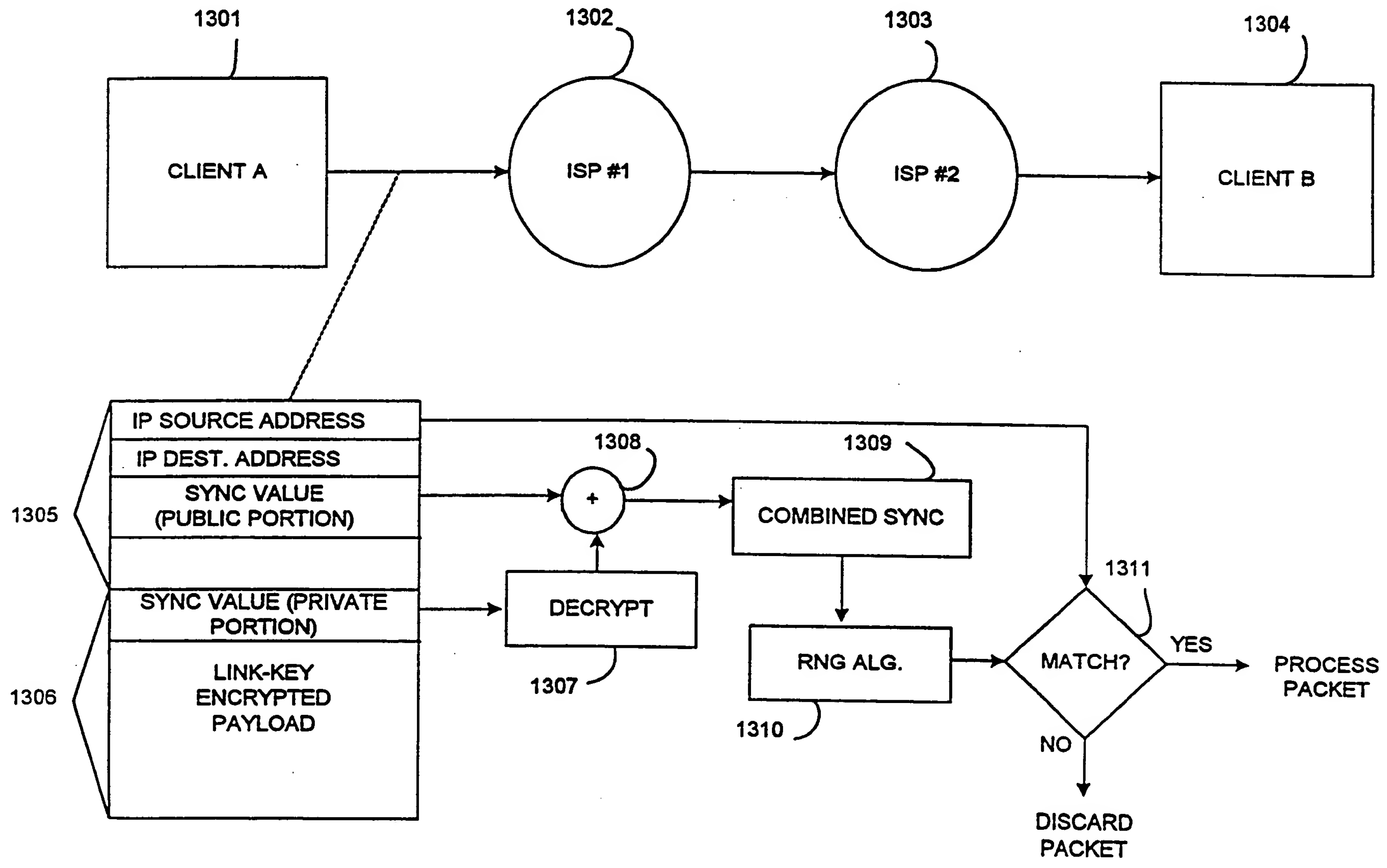

1301
1302
1303
1304
CLIENT A
ISP #1
ISP #2
CLIENT B
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1305
1306
1308
+
1309
COMBINED SYNC
DECRYPT
1307
RNG ALG.
1310
1311
MATCH?
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13

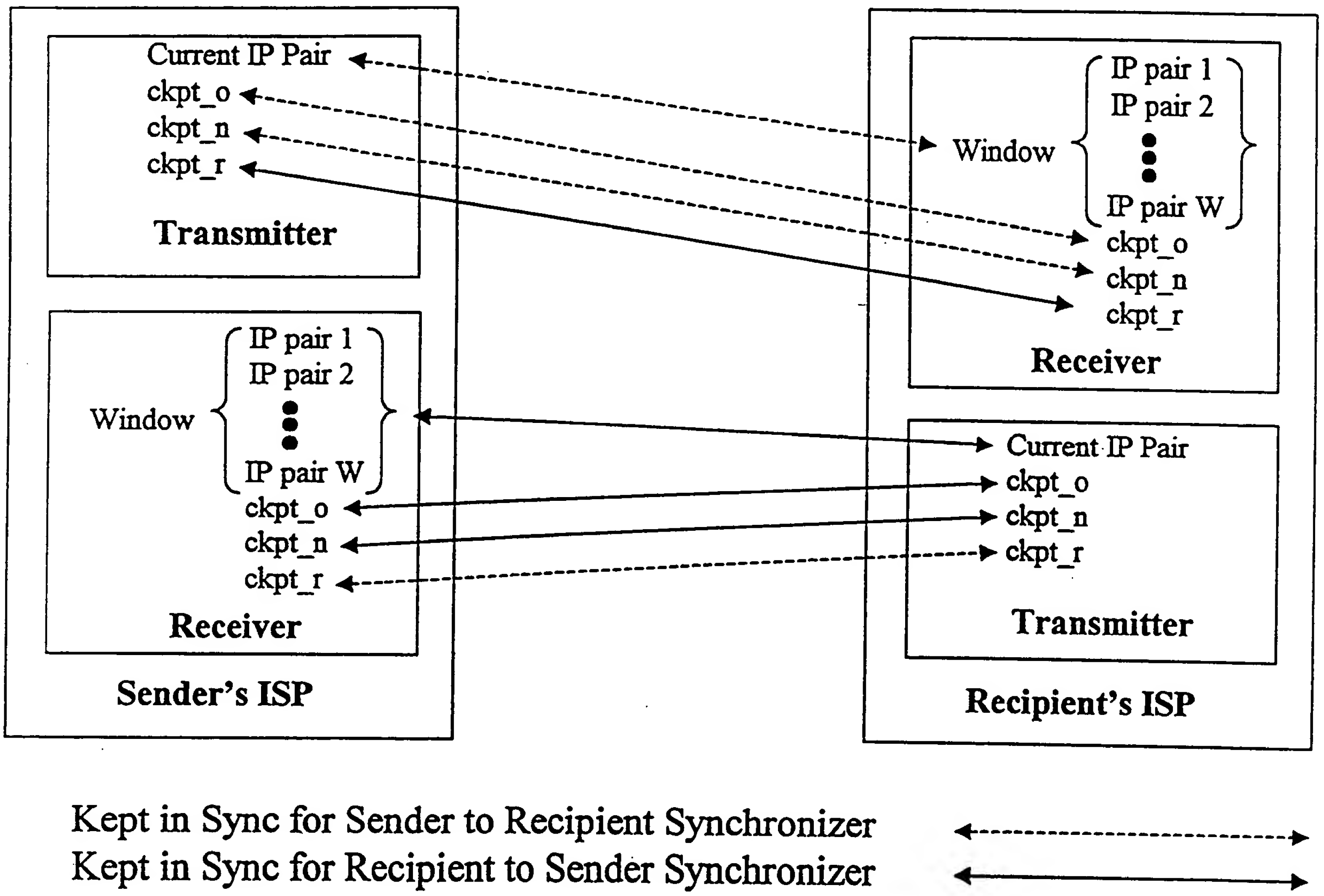
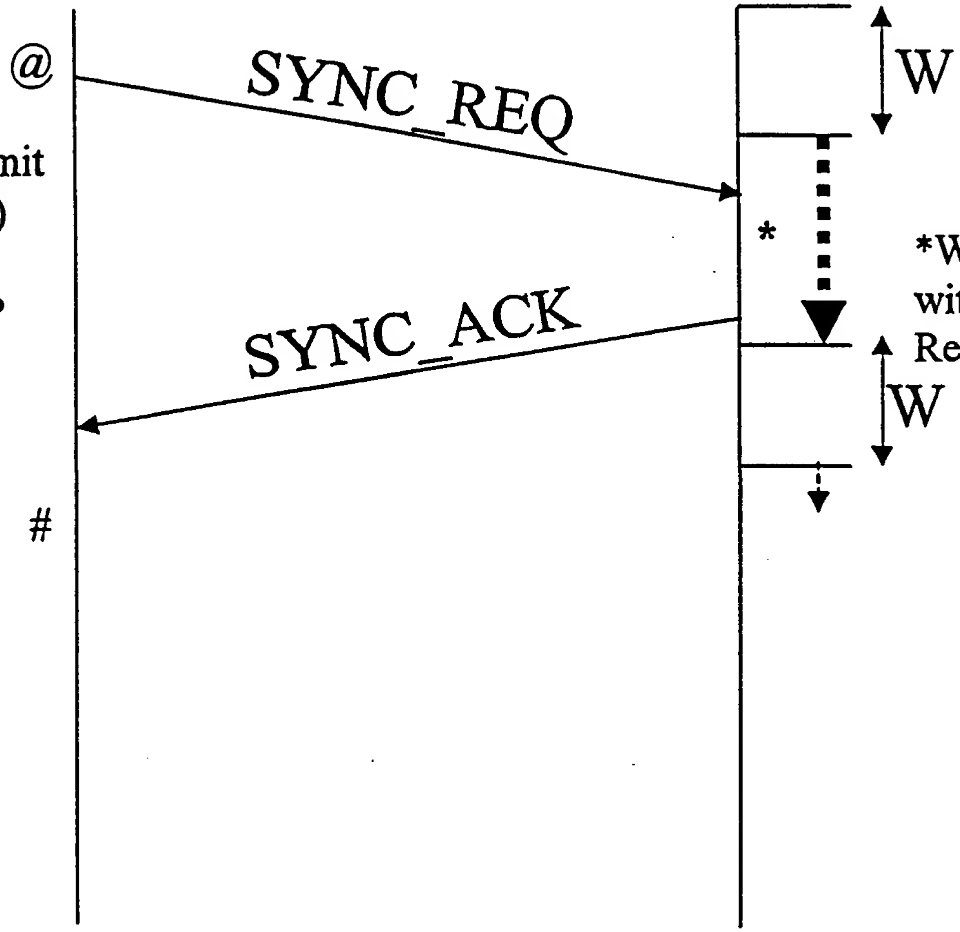

Current IP Pair
ckpt_o
ckpt_n
ckpt_r
Transmitter
Window
IP pair 1
IP pair 2
IP pair W
ckpt_o
ckpt_n
ckpt_r
Receiver
Sender's ISP
Window
IP pair 1
IP pair 2
IP pair W
ckpt_o
ckpt_n
ckpt_r
Receiver
Current IP Pair
ckpt_o
ckpt_n
ckpt_r
Transmitter
Recipient's ISP
Kept in Sync for Sender to Recipient Synchronizer
Kept in Sync for Recipient to Sender Synchronizer

FIG. 14



@ When Synchronization Begins Transmit (Retransmit Periodically Until ACKed) SYNC_REQ Using New Transmitter Checkpoint IP Pair ckpt_n and Generate New Receiver Response Checkpoint ckpt_r

When SYNC_ACK Arrives with Incoming Header = ckpt_r: Generate New Checkpoint IP Pair ckpt_n in Transmitter

*When SYNC_REQ Arrives with Incoming Header = Receiver's ckpt_n:

- Update Window
- Generate New Checkpoint IP Pair ckpt_n in Receiver
- Generate New Checkpoint IP Pair ckpt_r in Transmitter
- Transmit SYNC_ACK Using New Checkpoint IP Pair ckpt_r

FIG. 15

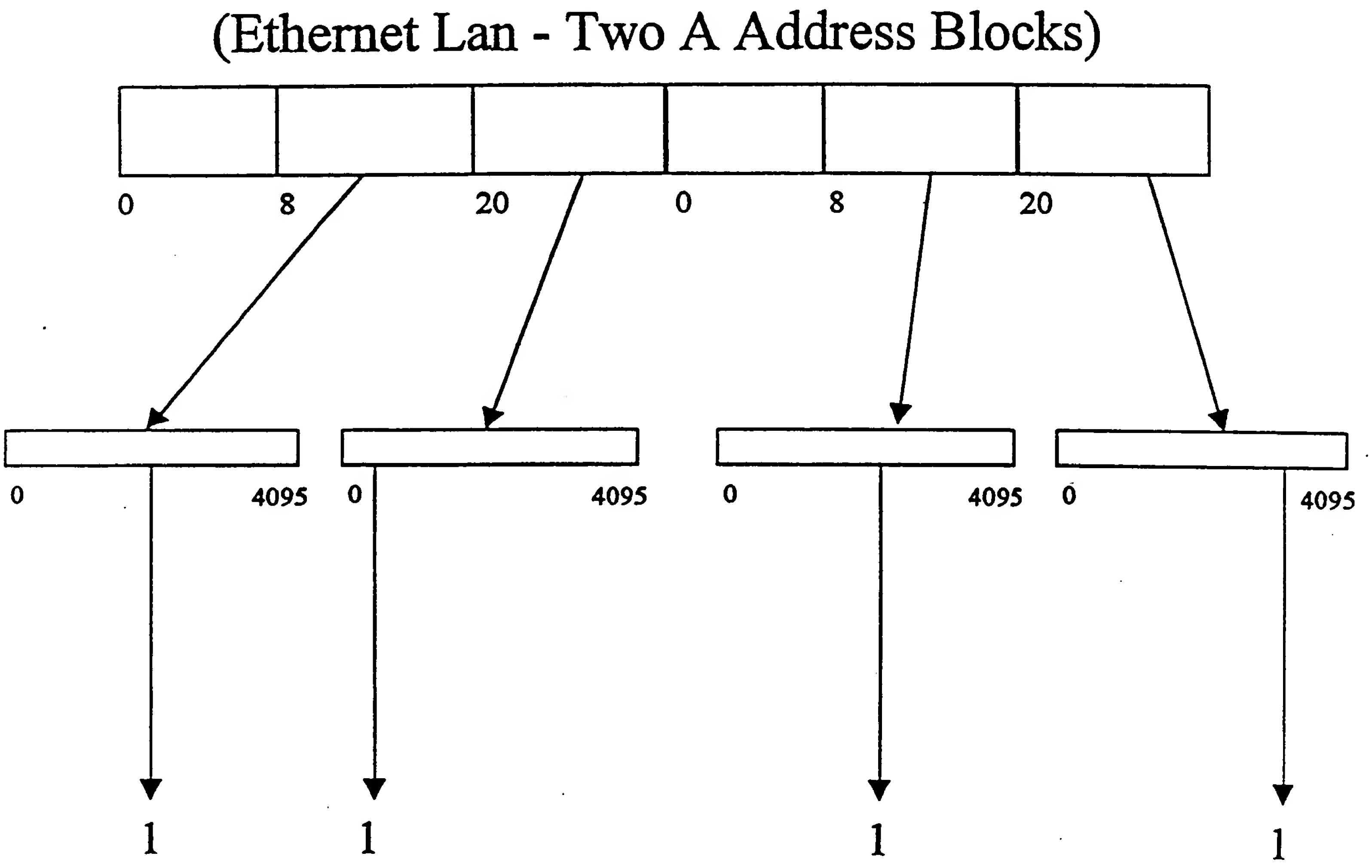

(Ethernet Lan - Two A Address Blocks)
0
8
20
0
8
20
0
4095
0
4095
0
4095
0
4095
1
1
1
1

FIG. 16

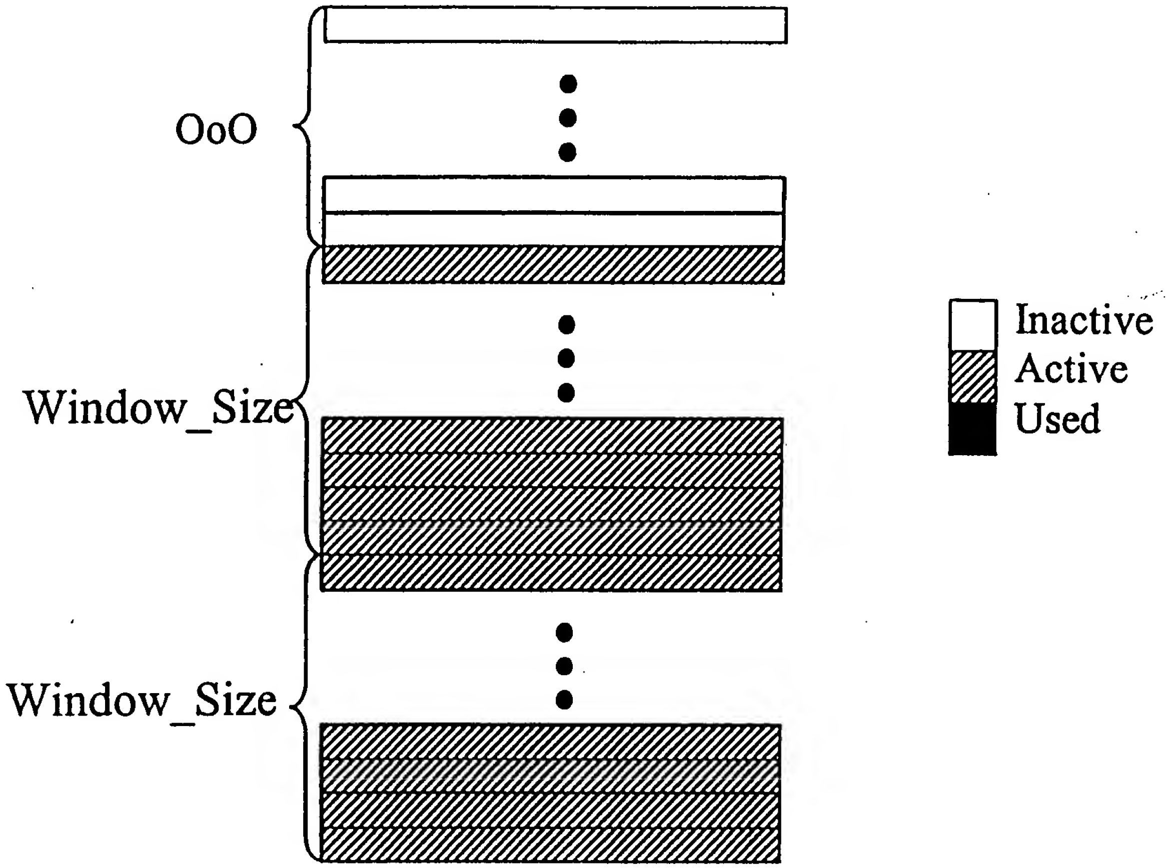


FIG. 17

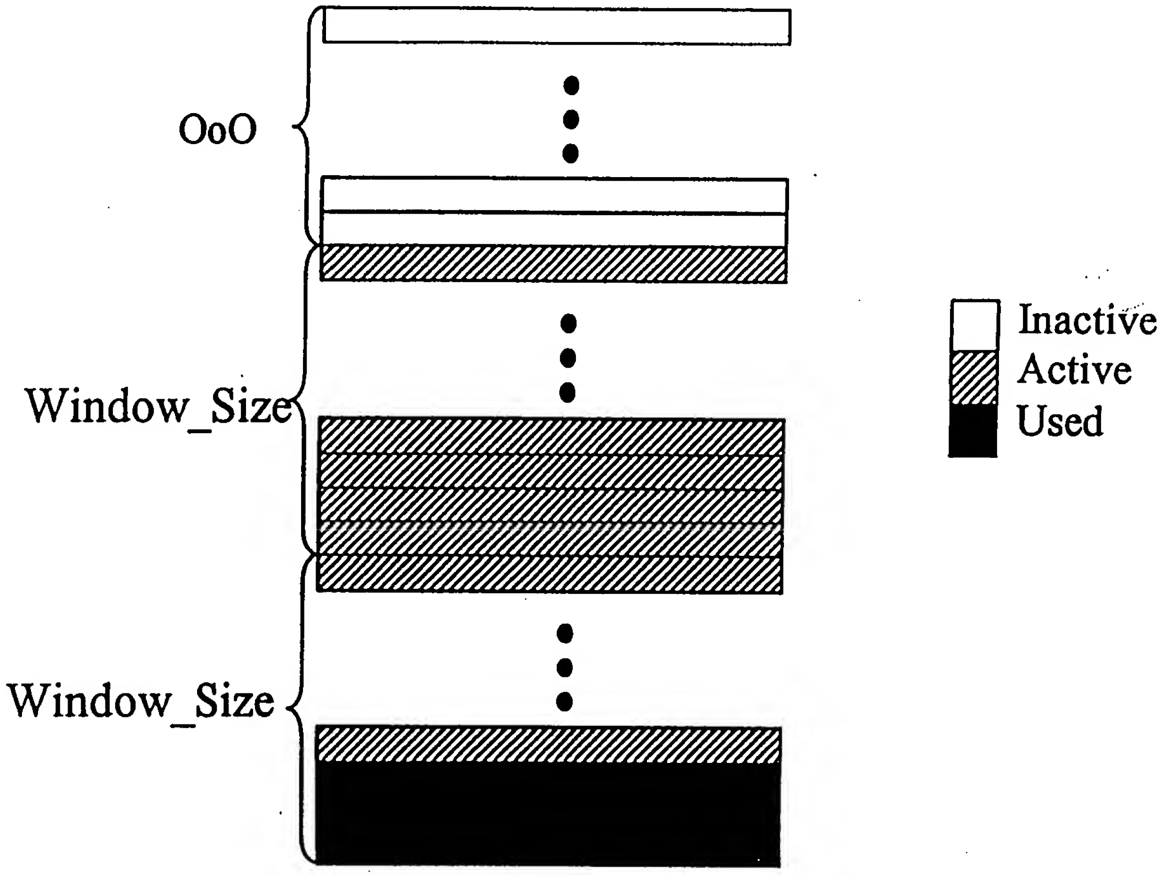


FIG. 18

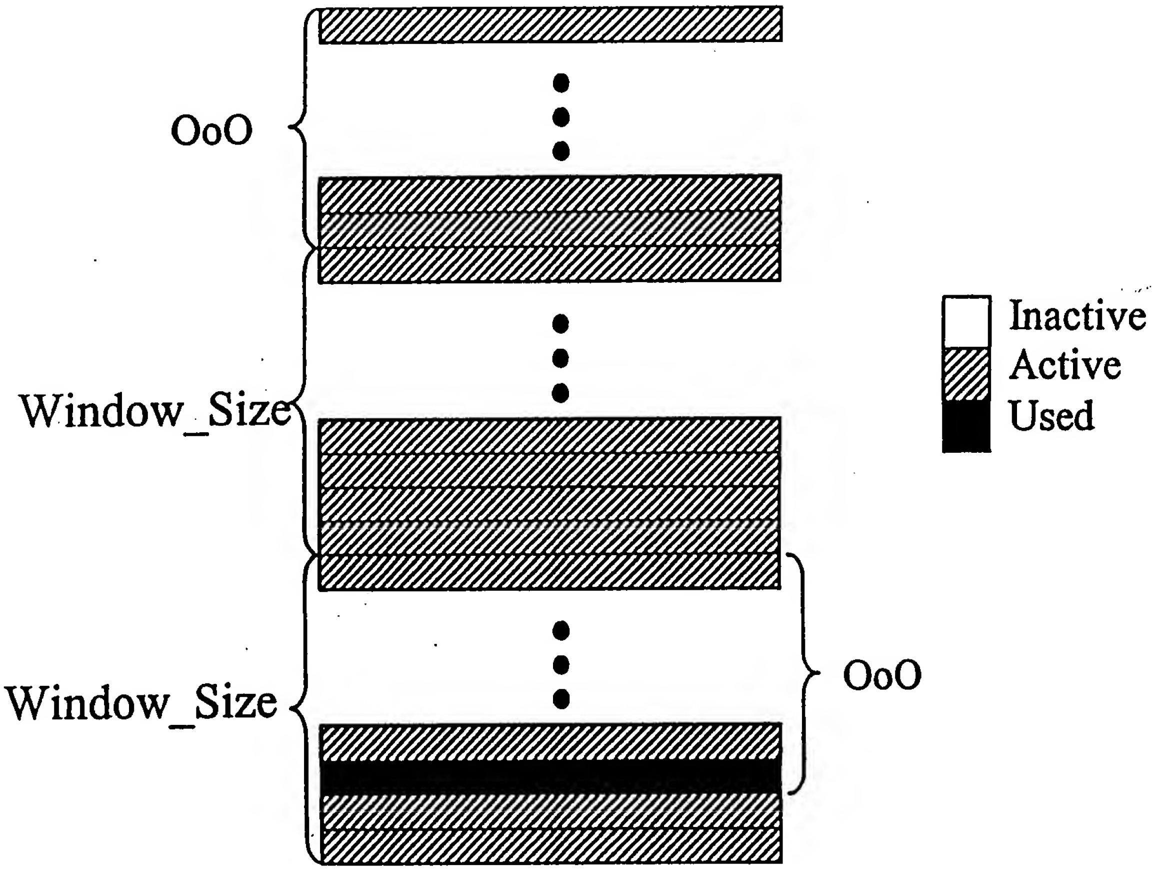


FIG. 19

FIG. 20

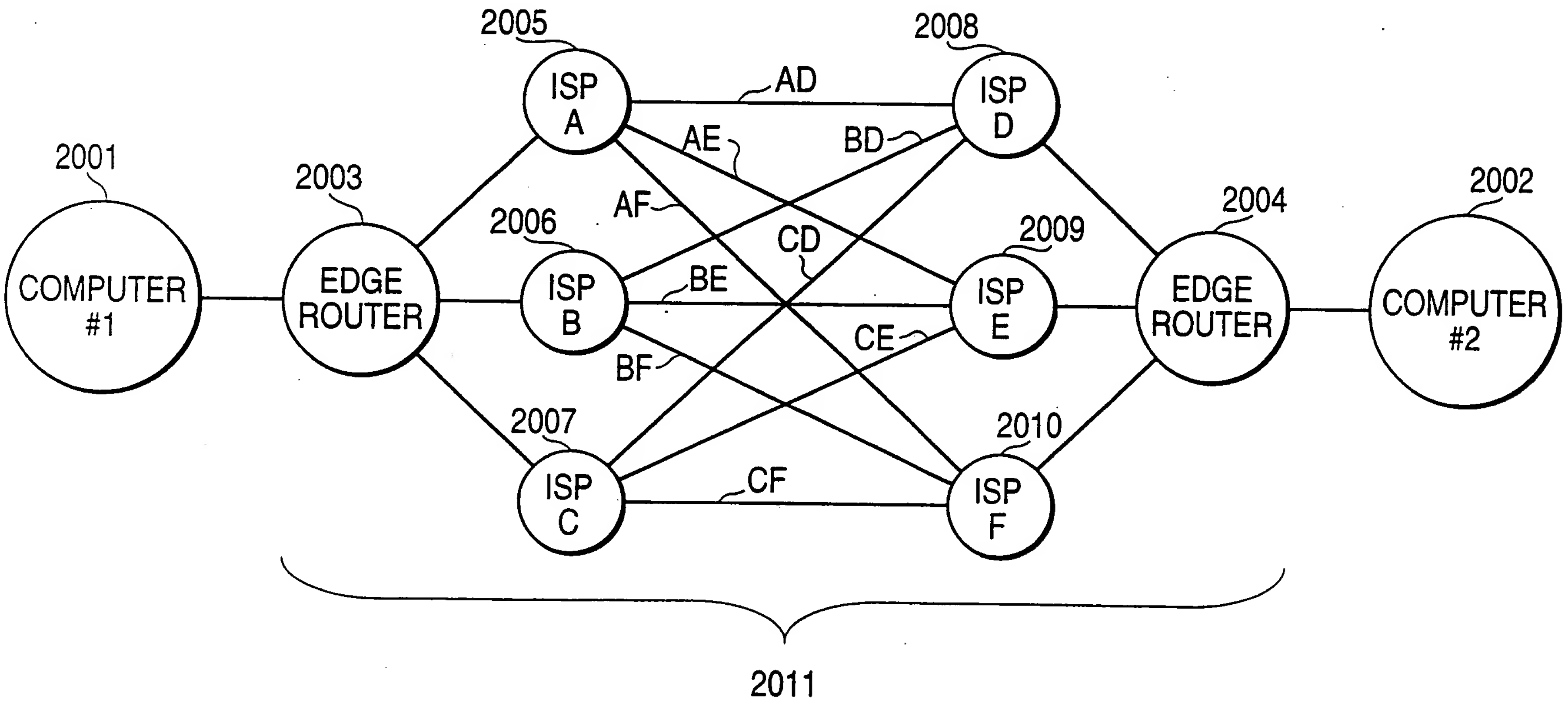
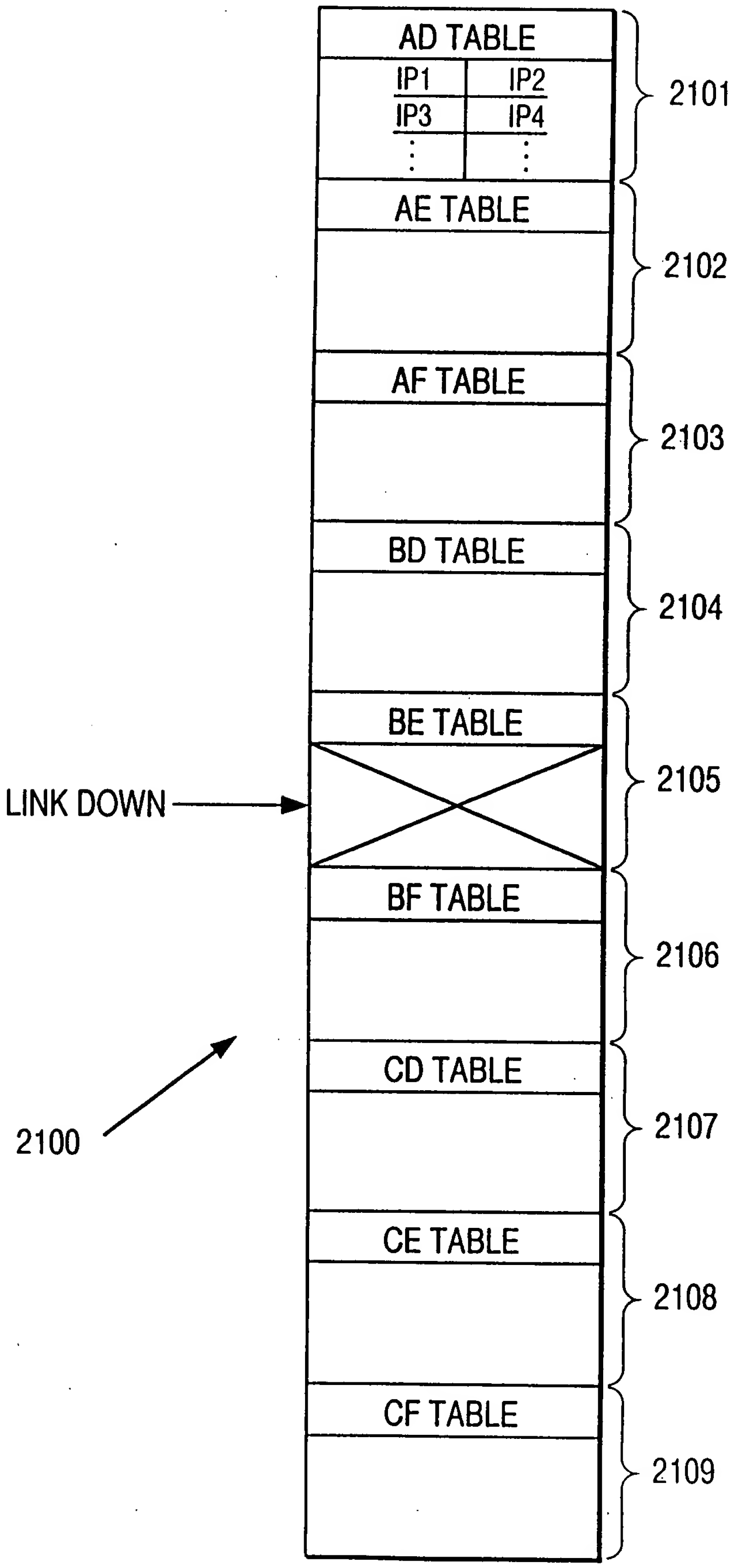

2001
COMPUTER #1
2003
EDGE ROUTER
2005
ISP A
2006
ISP B
2007
ISP C
AD
AE
AF
BD
BE
BF
CD
CE
CF
2008
ISP D
2009
ISP E
2010
ISP F
2004
EDGE ROUTER
2002
COMPUTER #2
2011

FIG. 21


AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
2105
LINK DOWN
BF TABLE
2106
CD TABLE
2107
2100
CE TABLE
2108
CF TABLE
2109

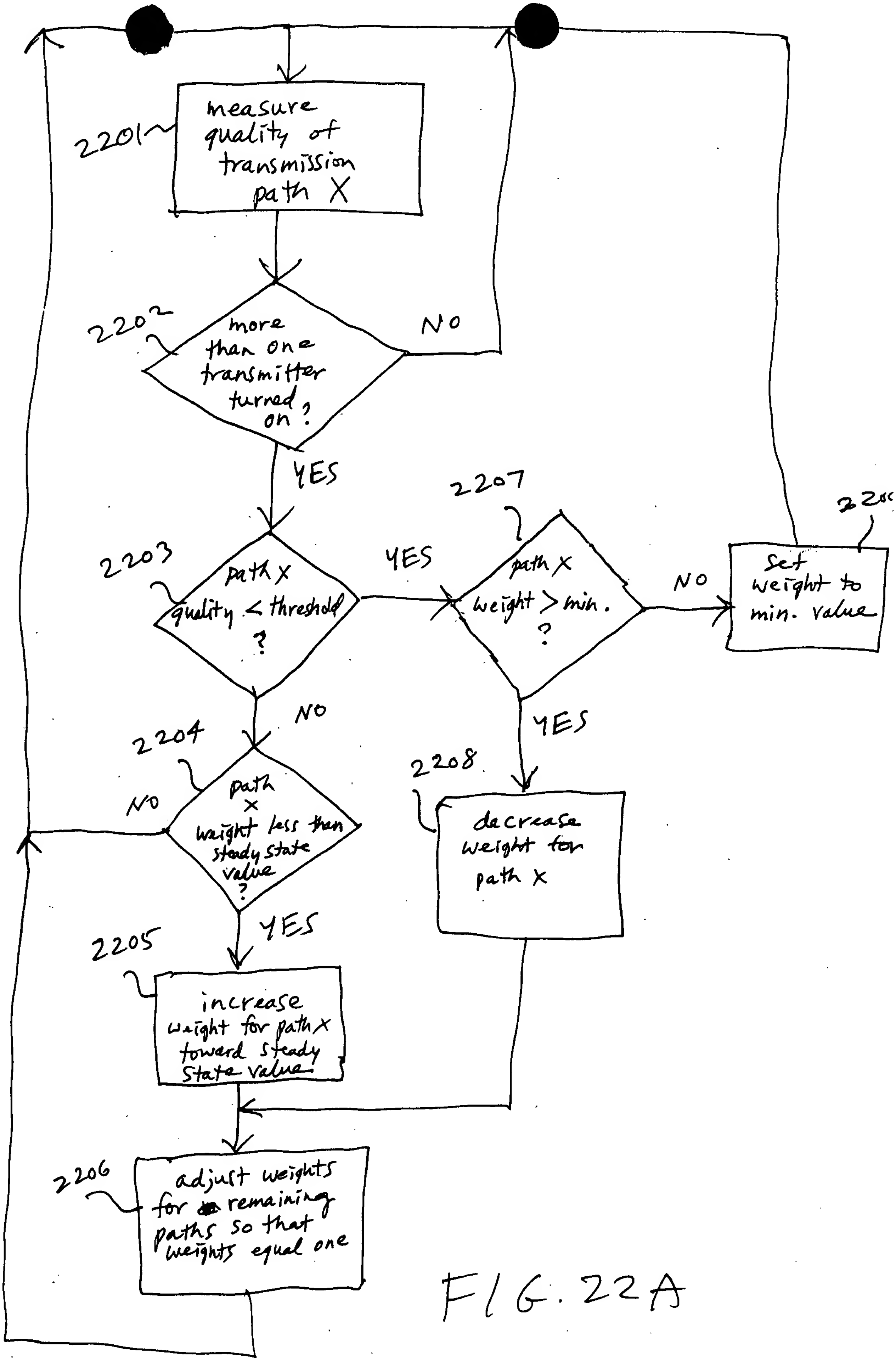

measure quality of transmission path X
2201
more than one transmitter turned on?
2202
NO
YES
path X quality < threshold ?
2203
YES
path X weight > min. ?
2207
NO
Set weight to min. value
220[illegible]
NO
path X weight less than steady state value ?
2204
NO
YES
increase weight for path X toward steady State value
2205
YES
decrease weight for path X
2208
adjust weights for remaining paths so that weights equal one
2206

FIG. 22A

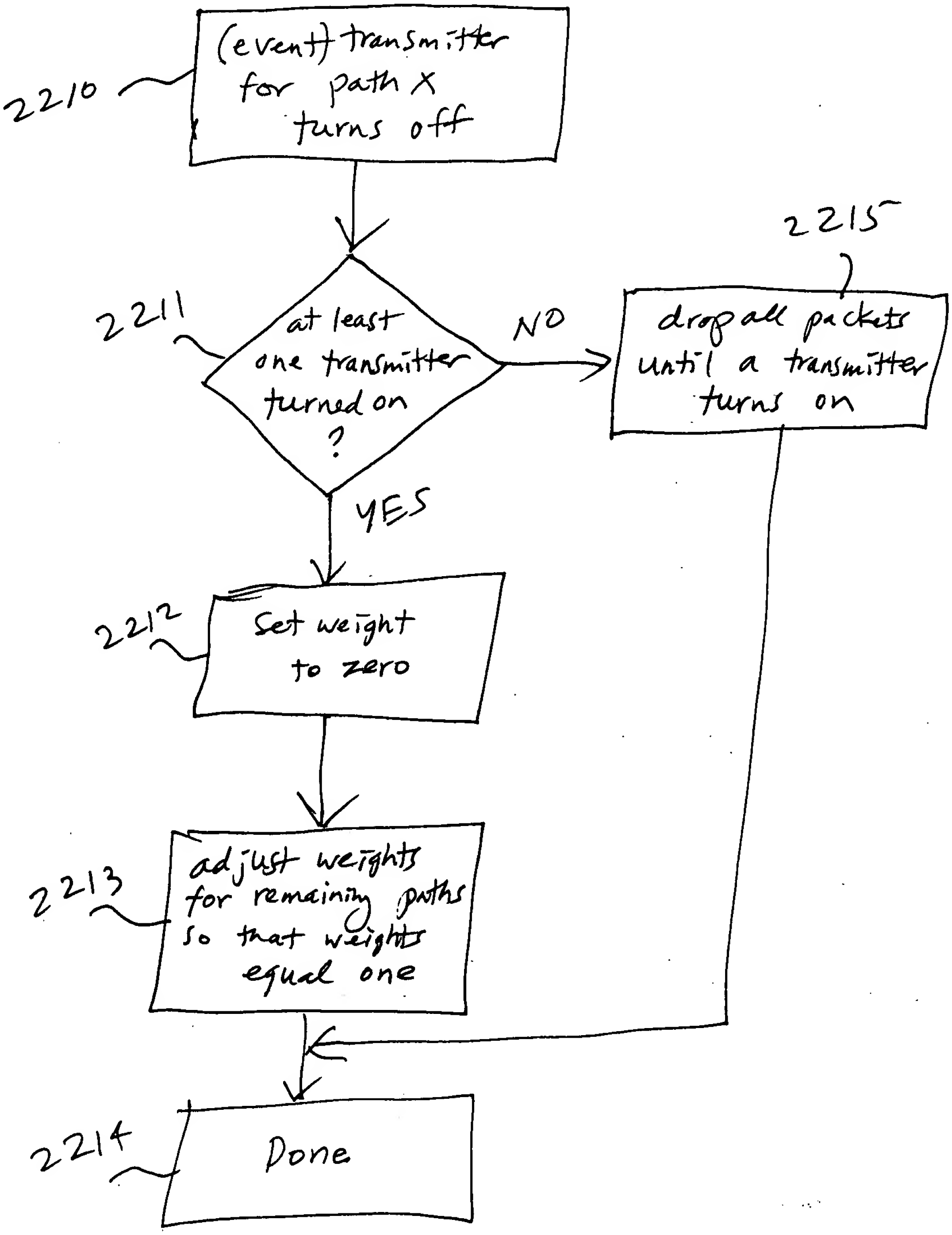

(event) transmitter for path X turns off
2210
at least one transmitter turned on ?
2211
NO
drop all packets until a transmitter turns on
2215
YES
Set weight to zero
2212
adjust weights for remaining paths so that weights equal one
2213
Done
2214

FIG. 22B

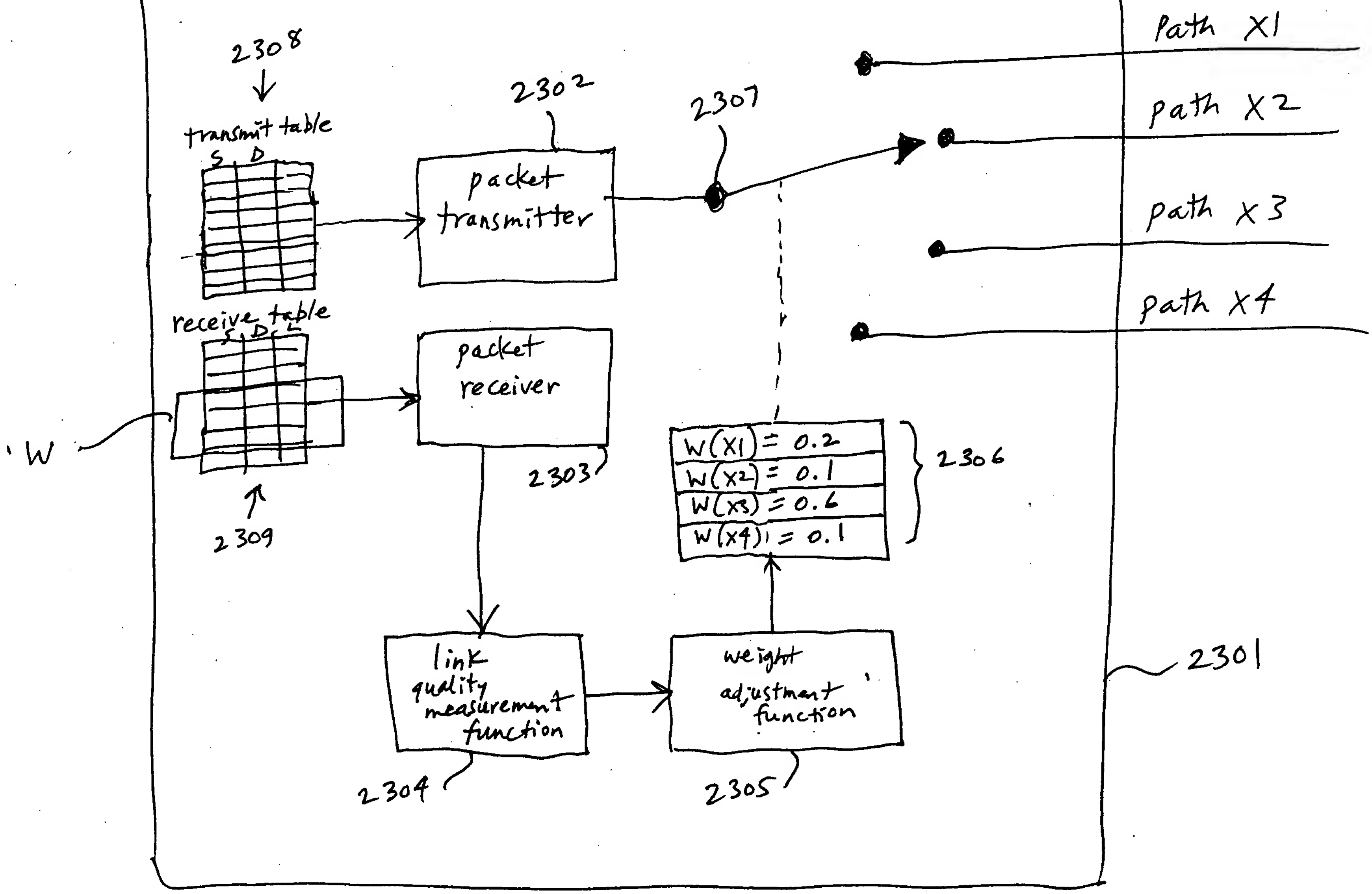

2308
transmit table
S D
2302
packet transmitter
2307
path X1
path X2
path X3
path X4
receive table
S D L
packet receiver
'W
2309
2303
W(X1) = 0.2
W(X2) = 0.1
W(X3) = 0.6
W(X4) = 0.1
2306
link quality measurement function
weight adjustment function
2304
2305
2301

FIG. 23

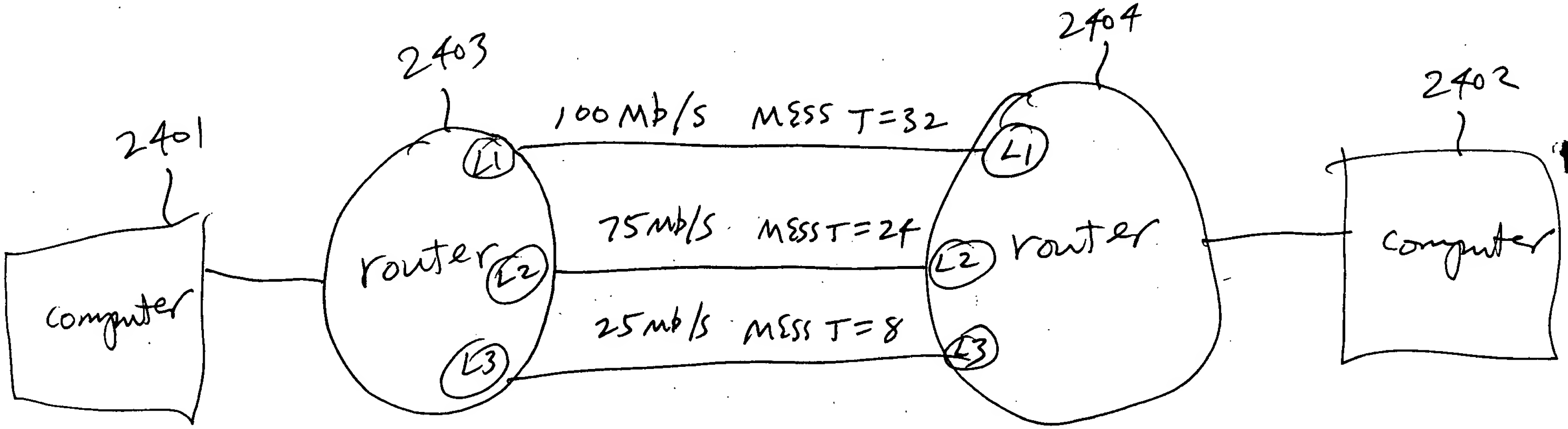

2401
computer
2403
router
L1
L2
L3
100Mb/s MSSS T=32
75Mb/s MSSS T=24
25Mb/s MSSS T=8
2404
router
L1
L2
L3
2402
computer

FIG. 24

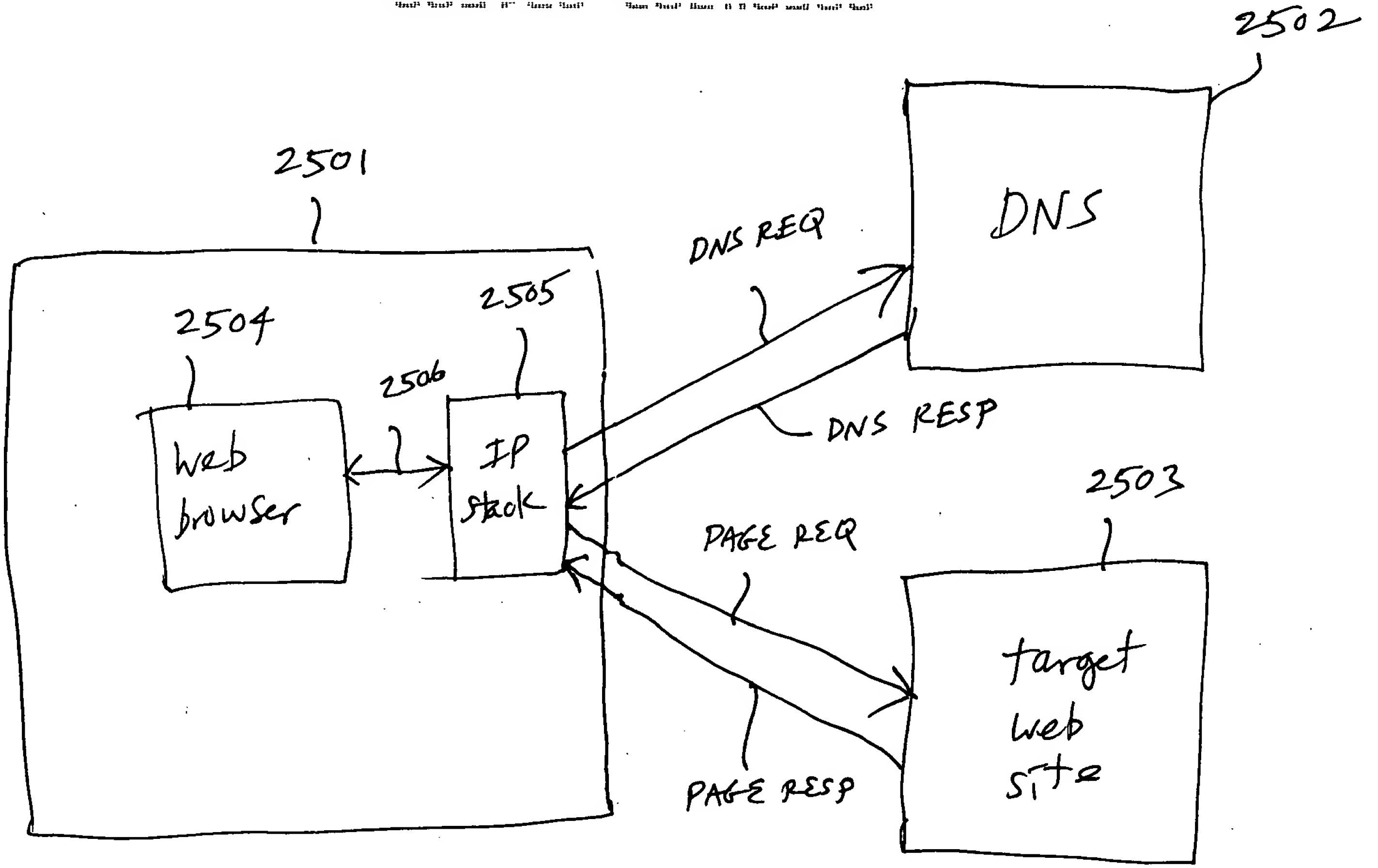


FIG. 25
(prior art)

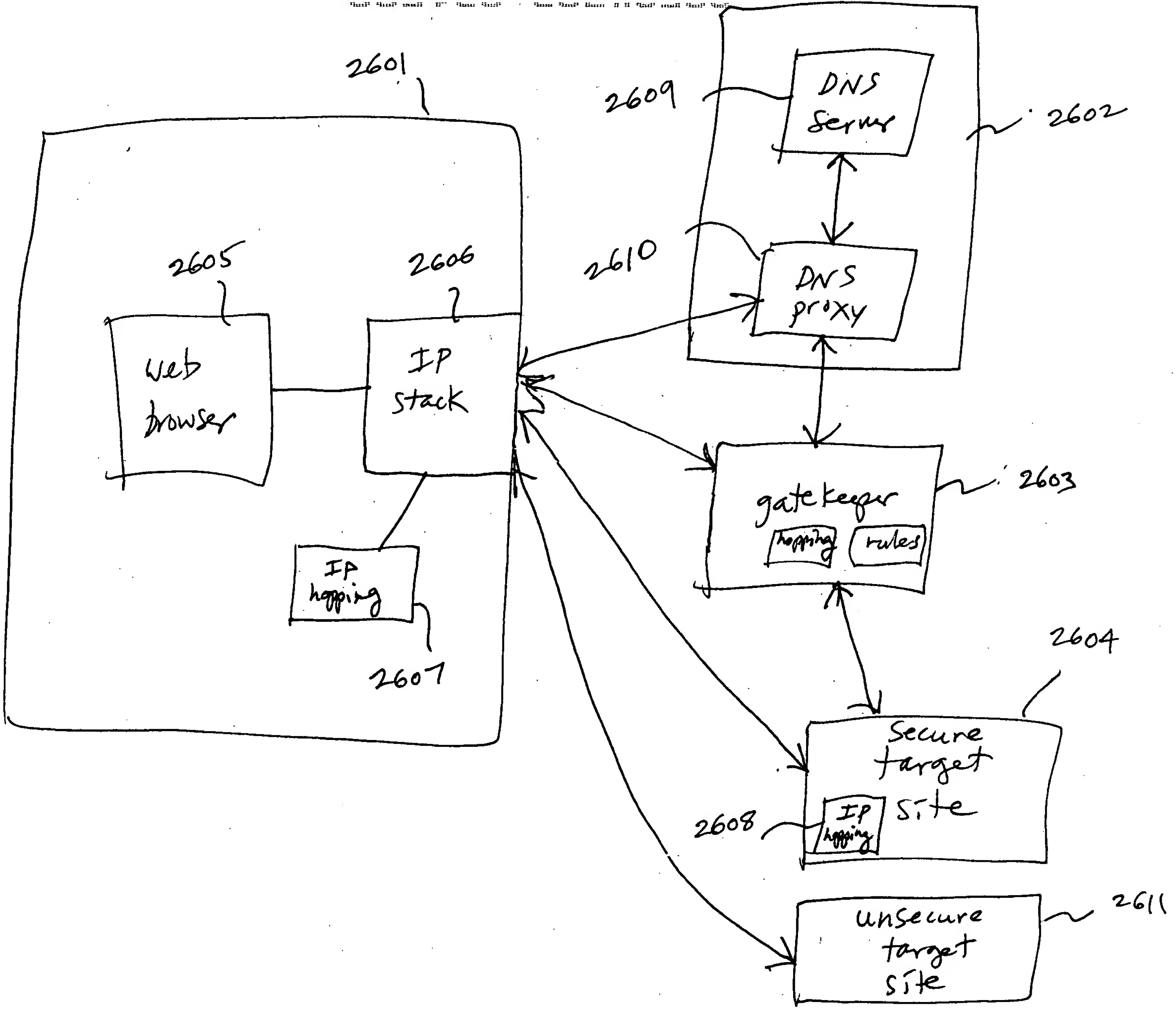

2601
2605
web browser
2606
IP stack
IP hopping
2607
2609
DNS Server
2602
2610
DNS proxy
gatekeeper
hopping
rules
2603
2604
Secure target site
IP hopping
2608
unsecure target site
2611

FIG. 26

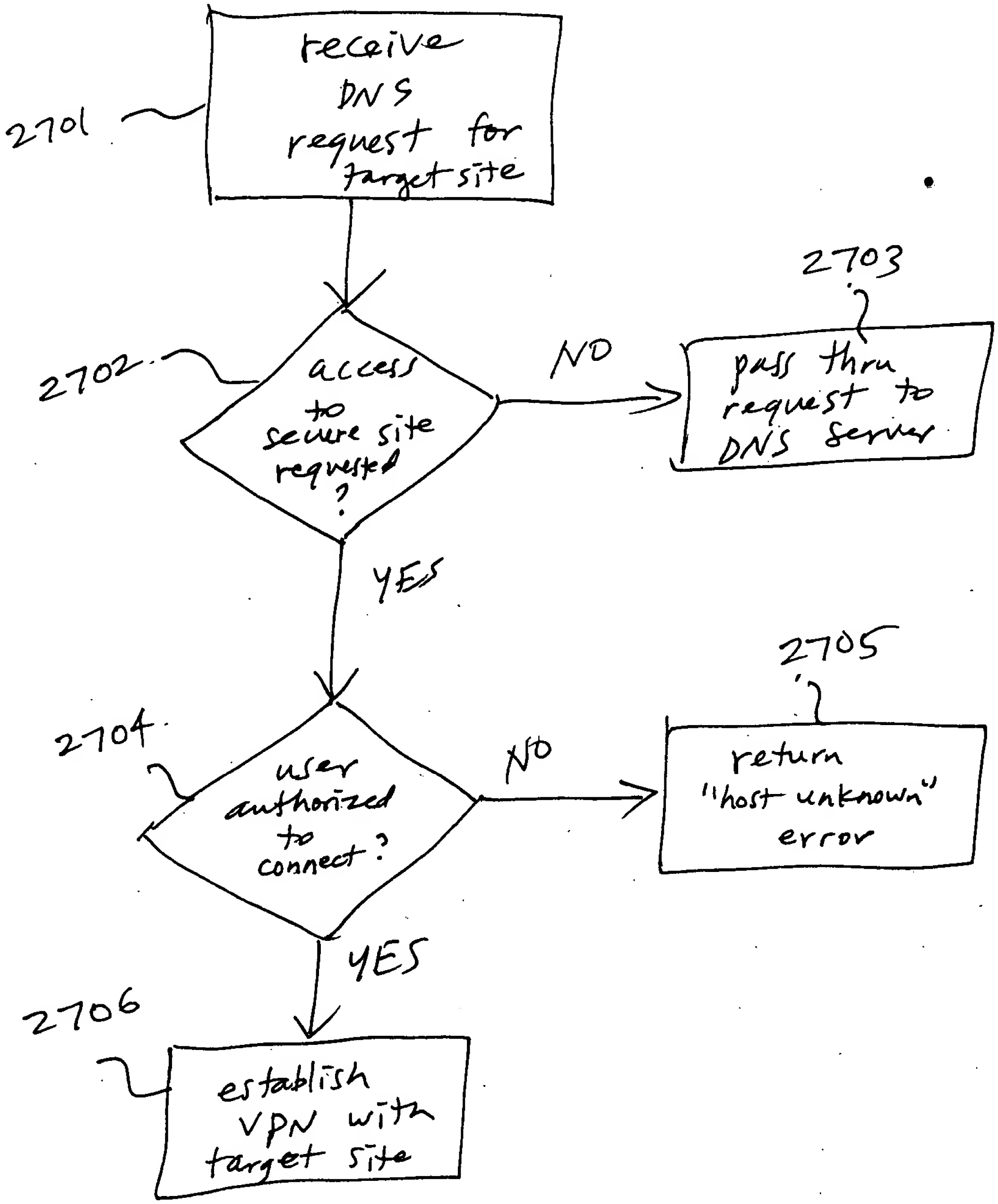

receive DNS request for target site
2701
access to secure site requested?
2702
NO
pass thru request to DNS server
2703
YES
user authorized to connect?
2704
NO
return "host unknown" error
2705
YES
establish VPN with target site
2706

FIG. 27

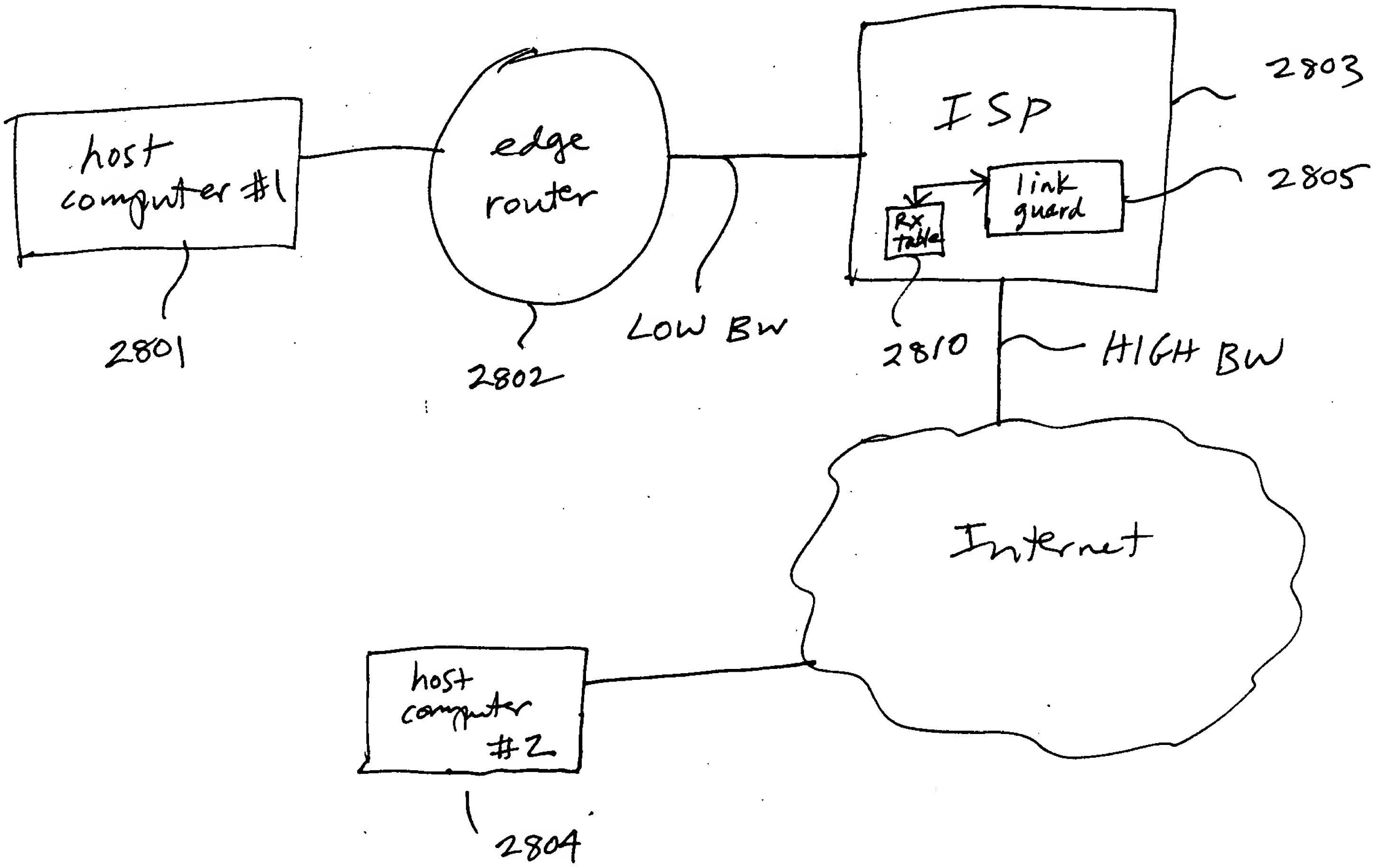

host computer #1
2801
edge router
2802
LOW BW
ISP
2803
link guard
2805
Rx Table
2810
HIGH BW
Internet
host computer #2
2804

FIG. 28

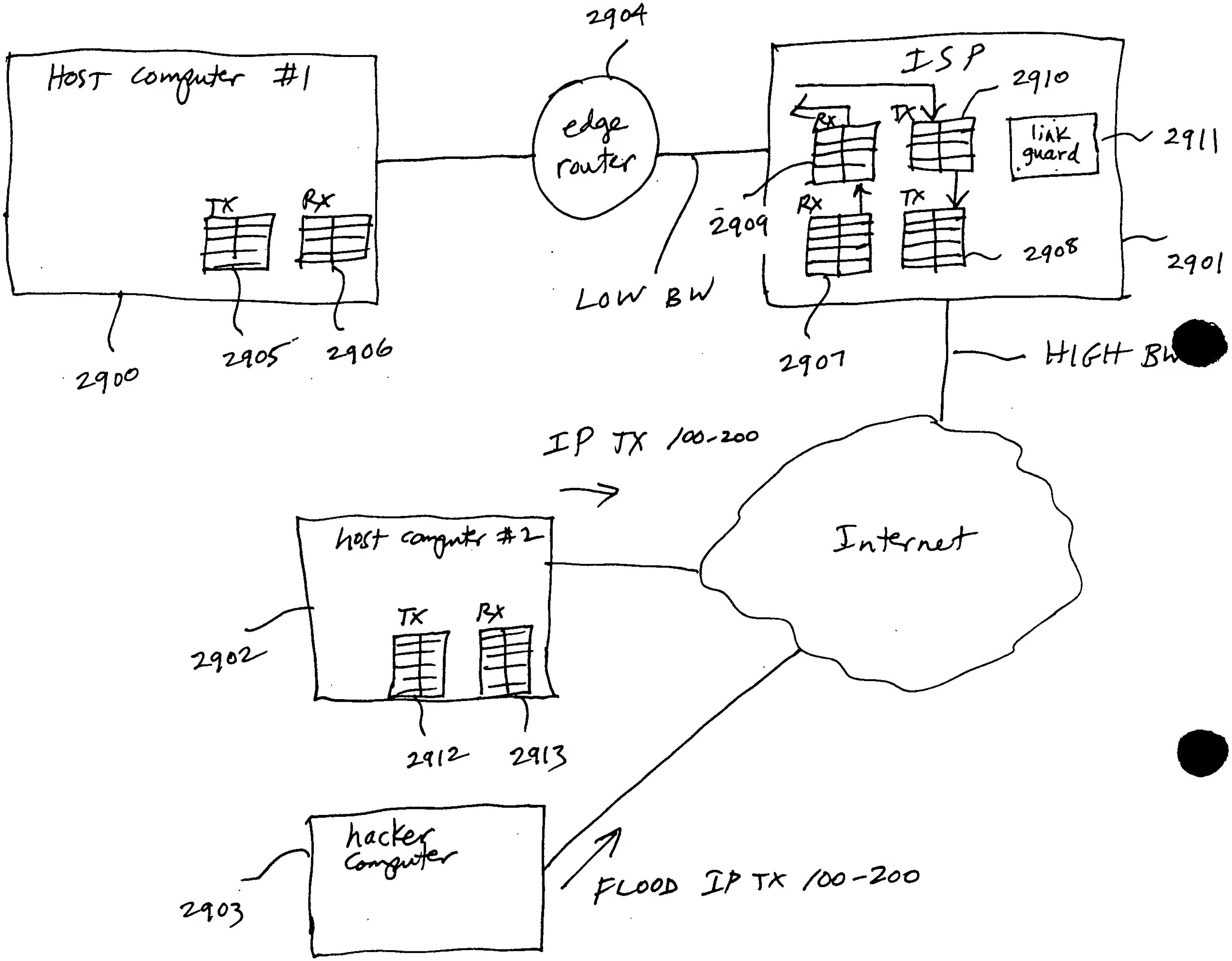

2904
Host computer #1
edge router
ISP
2910
link guard
2911
TX
RX
TX
RX
2905
2906
2900
2909
LOW BW
2907
2908
2901
HIGH BW
IP TX 100-200
Internet
host computer #2
TX
RX
2902
2912
2913
hacker computer
2903
FLOOD IP TX 100-200

FIG. 29

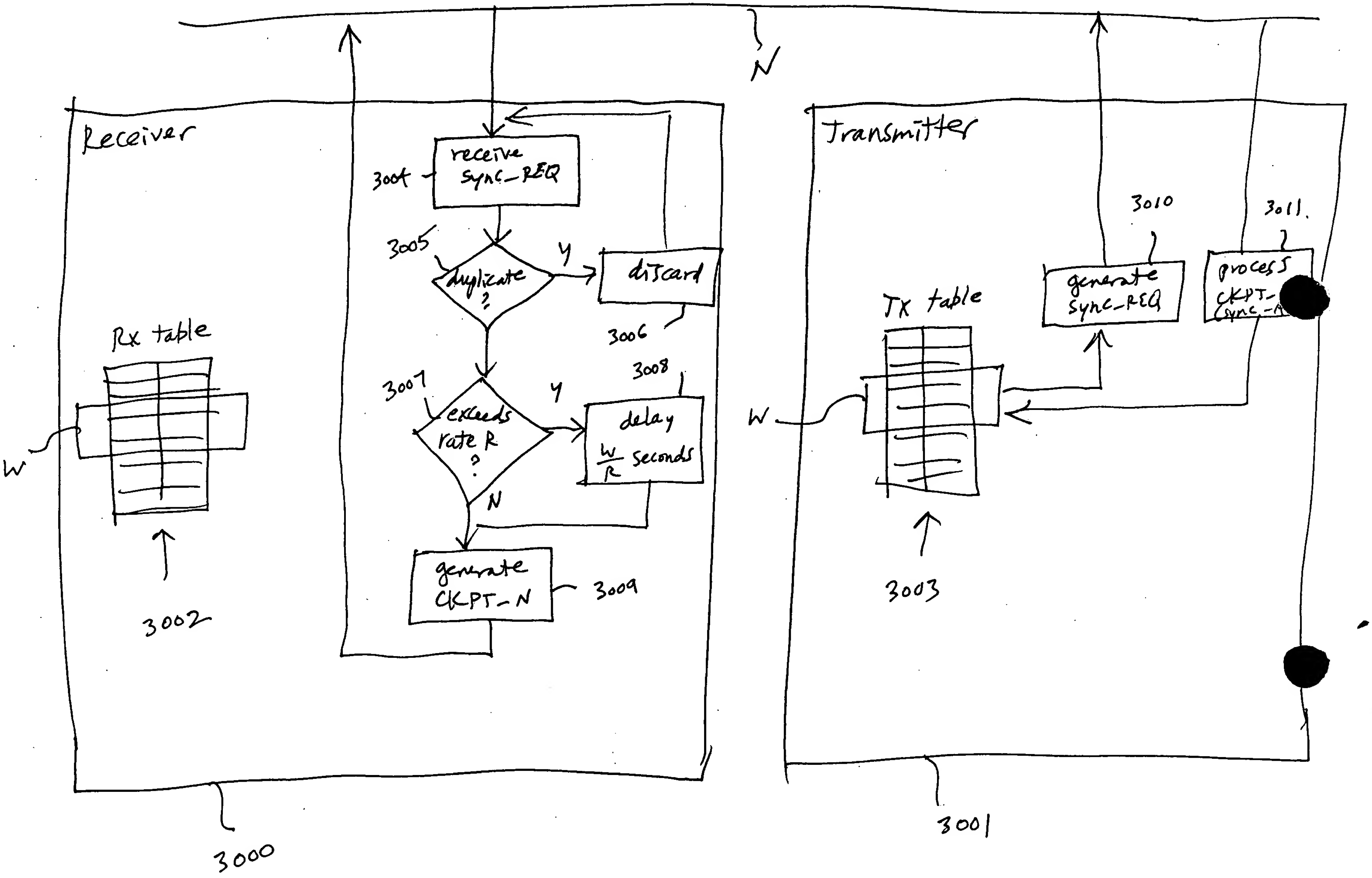

N
Receiver
receive sync_REQ
3004
3005
duplicate ?
Y
discard
3006
3007
exceeds rate R ?
Y
3008
delay $\frac{W}{R}$ seconds
N
generate CKPT_N
3009
Rx table
W
3002
3000
Transmitter
3010
3011.
generate sync_REQ
process CKPT_(sync_R
Tx table
W
3003
3001

FIG. 30

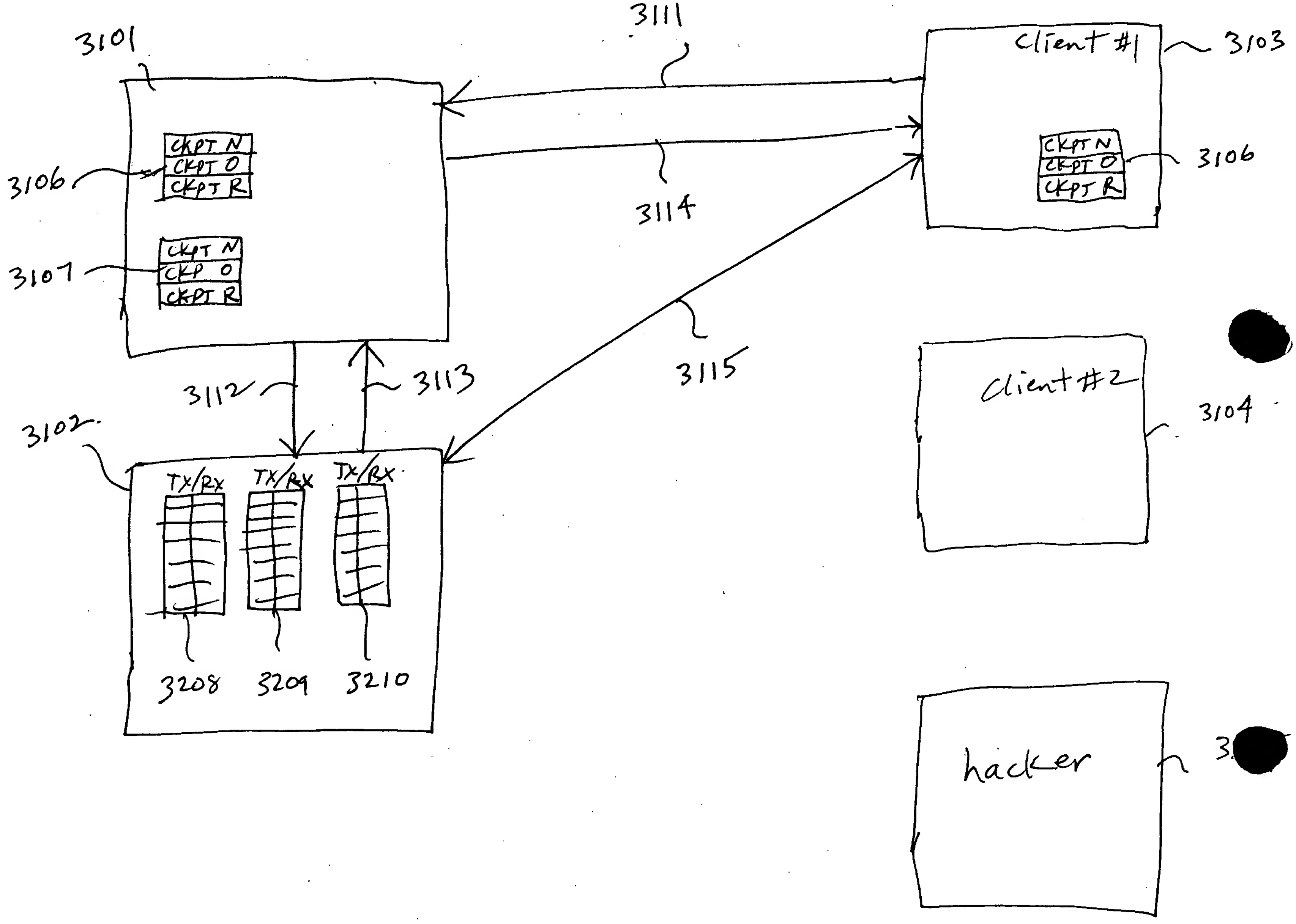

3101
3106
CKPT N
CKPT O
CKPT R
3107
CKPT N
CKP O
CKPT R
3111
3114
3115
3112
3113
3102
TX/RX
TX/RX
TX/RX
3208
3209
3210
Client #1
3103
CKPT N
CKPT O
CKPT R
3106
Client #2
3104
hacker
3

FIG. 31

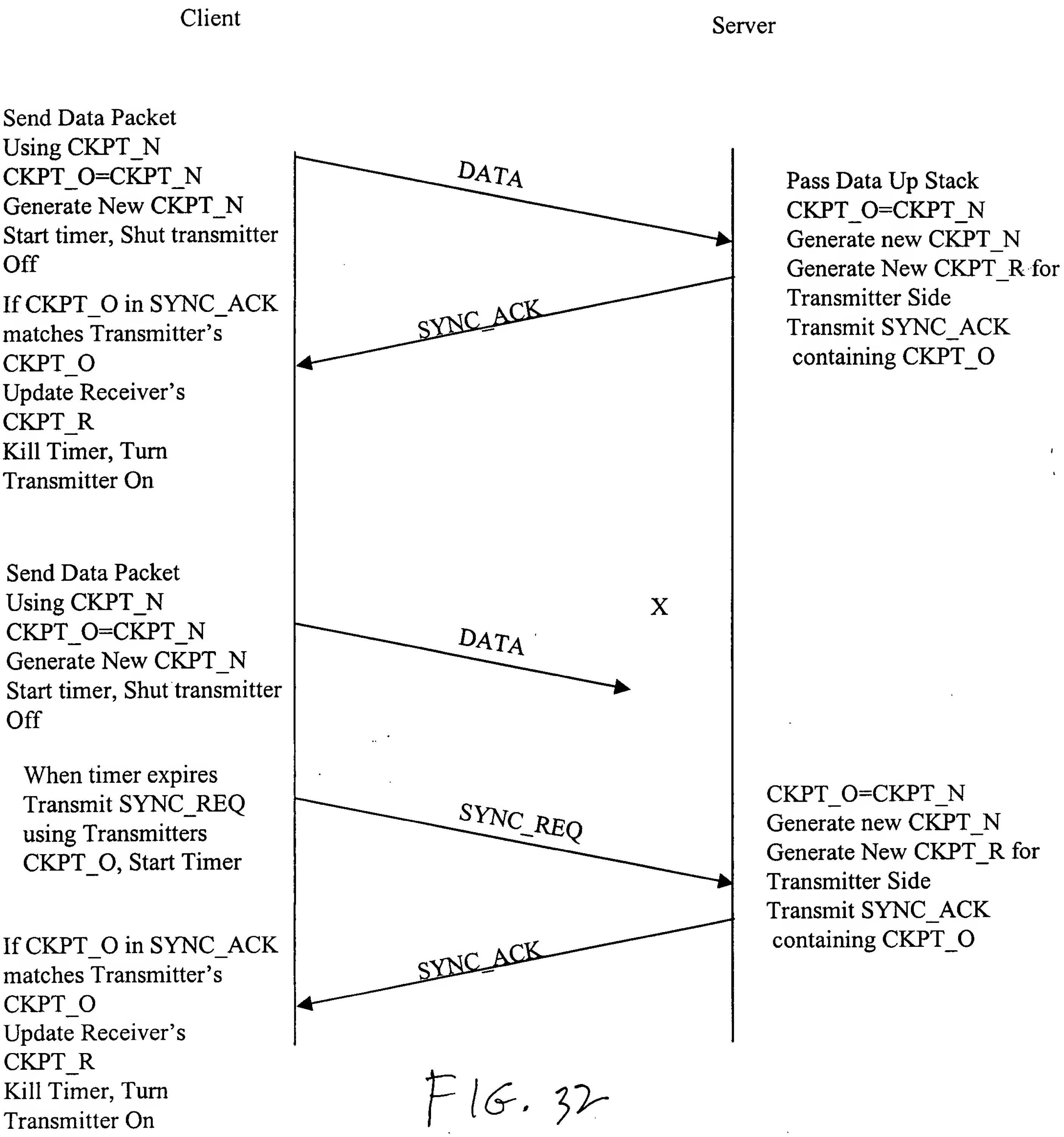

Client
Server
Send Data Packet
Using CKPT_N
CKPT_O=CKPT_N
Generate New CKPT_N
Start timer, Shut transmitter
Off
DATA
Pass Data Up Stack
CKPT_O=CKPT_N
Generate new CKPT_N
Generate New CKPT_R for
Transmitter Side
Transmit SYNC_ACK
containing CKPT_O
SYNC_ACK
If CKPT_O in SYNC_ACK
matches Transmitter's
CKPT_O
Update Receiver's
CKPT_R
Kill Timer, Turn
Transmitter On
Send Data Packet
Using CKPT_N
CKPT_O=CKPT_N
Generate New CKPT_N
Start timer, Shut transmitter
Off
DATA
X
When timer expires
Transmit SYNC_REQ
using Transmitters
CKPT_O, Start Timer
SYNC_REQ
CKPT_O=CKPT_N
Generate new CKPT_N
Generate New CKPT_R for
Transmitter Side
Transmit SYNC_ACK
containing CKPT_O
SYNC_ACK
If CKPT_O in SYNC_ACK
matches Transmitter's
CKPT_O
Update Receiver's
CKPT_R
Kill Timer, Turn
Transmitter On

FIG. 32